UNIVERSITY OF CALIFORNIA, SAN DIEGO

Modeling and Optimizing User Experience for Cloud Mobile 3D Applications

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

in

Electrical Engineering (Computer Engineering)

by

Yao Lu

Committee in charge:

Professor Sujit Dey, Chair
Professor Pamela Cosman
Professor Truong Nguyen
Professor Jurgen P. Schulze
Professor Geoffrey M. Voelker

2016

The Dissertation of Yao Lu is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Sujit Dey for his support as my advisor as well as the chair of my committee. Under his careful supervision and guidance, I learned a lot of technical knowledge very fast, developed brand new techniques solidly, improved my writing skills and presentation skills in a systematic way. His mentorship is the key to the success of my earning the degree.

I would also like to acknowledge my parents and family. They provided tremendous support and help in the whole career of my Ph.D. life, especially when I met difficulties. Without their support, it would be 3 times more difficult for me to fulfill the degree requirement in time.

I would also like to acknowledge my lab mates. Senior students such as Yao Liu helped me a lot preparing manuscripts for publication. Junior students such as Wenchuan Wei and Xueshi Hou worked closely with me together on a lot of cool techniques & systems. I sincerely appreciate all their effort.

Chapter 2, in part, is from the material as it appears in proceedings of IEEE ICNC 2014. Yao Lu; Yao Liu; Sujit Dey. and in IEEE Journal of Selected Topics in Signal Processing 2015. Yao Lu; Yao Liu; Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is from the material as it appears in proceedings of IEEE ICC 2015. Yao Lu; Yao Liu; Sujit Dey. and in Multimedia Tools and Applications 2016. Yao Lu; Yao Liu; Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is from the material as it appears in proceedings of IEEE ISM 2015. Yao Lu; Yao Liu; Sujit Dey. and in IEEE Journal on Emerging and Selected Topics in Circuits and Systems 2016. Yao Lu; Sujit Dey. The dissertation author was the primary investigator and author of this paper.

VITA

2011	Visiting Scholar, Stanford University
2012	Bachelor of Engineering, Tsinghua University
2014	Master of Science, University of California, San Diego
2012–2016	Research Assistant, University of California, San Diego
2016	Doctor of Philosophy, University of California, San Diego

PUBLICATIONS

Xuan Dong, Guan Wang, Yi (Amy) Pang, Weixin Li, Jiangtao (Gene) Wen, Wei Meng and Yao Lu, "Fast Efficient Algorithm for enhancement of low lighting video" In Proceedings of IEEE International Conference on Multimedia and Expo (ICME), pp 1-6, Jul. 2011, Barcelona

Qiang Ning, Kan Chen, Li Yi, Chuchu Fan, Yao Lu and Jiangtao Wen, "Image Super-Resolution via Analysis Sparse Prior" IEEE Signal Processing Letters, Volume 20, Issue 4, pp 399-402, Jan. 2013

Tong Shen, Yao Lu, Ziyu Wen, Linxi Zou, Yucong Chen and Jiangtao Wen, "Ultra Fast H.264/AVC to HEVC Transcoder" In Proceedings of IEEE Data Compression Conference (DCC), pp 241-250, Mar. 2013, Snowbird

Jiangtao Wen, Shunyao Li, Yao Lu, Meiyuan Fang, Xuan Dong, Huiwen Chang and Pin Tao, "Cross Segment Decoding for Improved Quality of Experience for Video Applications" In Proceedings of IEEE Data Compression Conference (DCC), pp 231-240, Mar. 2013, Snowbird

Jiangtao Wen, Bohan Li, Shunyao Li, Yao Lu and Pin Tao, "Cross Segment Decoding of HEVC for Network Video Applications" In Proceedings of IEEE Packet Video Workshop, pp 1-8, Dec. 2013, San Jose

Sujit Dey, Yao Liu, Shaoxuan Wang and Yao Lu, "Addressing Response Time of Cloud-based Mobile Applications" In Proceedings of ACM International Workshop on Mobile Cloud Computing & Networking, pp 3-10, Jul. 2013, Bangalore

Yao Lu, Yao Liu and Sujit Dey, "Enhancing Cloud Mobile 3D Display Gaming User Experience by Asymmetric Graphics Rendering" In Proceedings of IEEE International

Conference on Computing, Networking and Communications (ICNC), pp 368-374, Feb. 2014, Honolulu

Yao Lu, Yao Liu and Sujit Dey, "Optimizing Cloud Mobile 3D Display Gaming User Experience by Asymmetric Object Of Interest Rendering" In Proceedings of IEEE International Conference on Communication (ICC), pp 6842-6848, Jun. 2015, London

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

Wenchuan Wei, Yao Lu, Cathrine Printz and Sujit Dey, "Motion Data Alignment and Real-Time Guidance in Cloud-Based Virtual Training System" In Proceedings of Wireless Health (WH), p. 13, Oct. 2015, Bethesda

Dennis Shen, Yao Lu and Sujit Dey, "Motion Data Alignment For Real-Time Guidance in Avatar Based Physical Therapy Training System" In Proceedings of IEEE International Conference on E-health Networking, Application & Services (Healthcom), Oct. 2015, Boston

Yao Liu, Sujit Dey and Yao Lu, "Enhancing Video Encoding for Cloud Gaming Using Rendering Information." IEEE Transactions on Circuits and Systems for Video Technology, Volume 25, Issue 12, pp 1960-1974, Dec. 2015

Yao Lu, Yao Liu and Sujit Dey, "A Joint Asymmetric Graphics Rendering and Video Encoding Approach for Optimizing Cloud Mobile 3D Display Gaming User Experience" In Proceedings of IEEE International Symposium on Multimedia (ISM), Dec. 2015, Miami

Yao Lu, Yao Liu and Sujit Dey, "Asymmetric and Selective Object Rendering for Optimized Cloud Mobile 3D Display Gaming User Experience" to appear in Multimedia Tools and Applications

Yao Lu and Sujit Dey, "JAVRE: A Joint Asymmetric Video Rendering and Encoding Approach to Enable Optimized Cloud Mobile 3D Virtual Immersive User Experience" to appear in IEEE Journal on Emerging and Selected Topics in Circuits and Systems

Ziyu Wen, Bichuan Guo, Jiashuo Liu, Jisheng Li, Yao Lu and Jiangtao Wen,"Novel 3D-WPP Algorithms for Parallel HEVC Encoding" In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar. 2016, Shanghai

Xueshi Hou, Yao Lu and Sujit Dey,"A Novel Hyper-cast Approach to Enable Cloud-based

Virtual Space Applications" to appear in Proceedings of IEEE International Symposium on Multimedia (ISM), Dec. 2016, San Jose

FIELDS OF STUDY

Major Field: Engineering

Studies in Electrical Engineering (Computer Engineering)
Professor Sujit Dey

ABSTRACT OF THE DISSERTATION

Modeling and Optimizing User Experience for Cloud Mobile 3D Applications

by

Yao Lu

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California, San Diego, 2016

Professor Sujit Dey, Chair

Cloud gaming architecture has been proposed and deployed in large-scale commercial systems in recent years. It renders and encodes video views on cloud servers, with the resulting video streamed through network to end devices. This approach has the advantage of relieving high computation, power and storage requirements of gaming from end devices. The challenge therefore shifts to streaming high quality video with low latency through networks with fluctuating conditions. In this thesis, we extend the basic cloud gaming architecture in three aspects. First, we consider not only gaming as target application, but also consider much more virtual immersive applications, such as

virtual classroom and virtual art gallery, etc. Second, with the development of ubiquitous wireless network for mobile devices, we specifically consider streaming the video through mobile wireless network. Third, with the growing popularity of mobile auto-stereoscopic 3D displays, we consider capture, record and stream two views (left and right views) in the virtual world so that it can render 3D views on mobile 3D displays. To conclude, we term our extended architecture as cloud mobile 3D display virtual immersive application architecture. Because 3D videos contain two views that doubles video bit rate requirement for the same quality versus 2D video, and also because mobile wireless network is much more fluctuating than wired and WiFi network, streaming high quality 3D video with low latency becomes much more challenging. In this thesis, based on prior research results that show human brain can compensate perceived video quality automatically when one of the views is of inferior quality, we propose asymmetric rendering where we tune graphics rendering quality to be asymmetric for the two views. We develop both user experience models and bit rate models by applying different rendering settings and conducting subjective tests. We further develop optimization algorithms which use the above two models to automatically decide the optimal rendering settings for left view and right view to ensure the best user experience given the network conditions. Experiments conducted using real 4G-LTE network profiles on commercial cloud service with different genres of applications demonstrate significant improvement in user experience when the proposed asymmetric rendering techniques are applied.

Chapter 1

Introduction

In this chapter, first, the background of cloud gaming technology will be introduced. How this technology has developed and evolved will be discussed and its advantages & disadvantages will be compared. Next, we will describe an overview of our extension to cloud mobile 3D display virtual immersive application architecture. System architecture will be provided and system design aspects will be discussed. We will then list the related work and existing commercial systems in this area. After that, we briefly introduce the novel technique of asymmetric graphics rendering that we propose in this thesis. Finally the organization of the thesis will be provided.

1.1 Background and System Architecture

Traditionally, before cloud gaming architecture [1] is proposed, people download from the Internet or buy a disc from local store to obtain a digital game. They then install the game on their desktop computer or laptop computer and every time launches it locally to play the game. The game utilizes local computing resources such as CPU and GPU to execute game logic and render the view. Some multi-user games will connect to the Internet and exchange some data through the network with very low bit rate. With this traditional architecture, gamers need to buy very high end computer with powerful CPU and GPU in order to play the very high end 3D games. In addition, it also requires

decent storage space for texture maps and mesh data for the scenes and objects. Besides, for developers, each game needs to be developed multiple times to support different hardware and operating systems platforms. It requires even much more effort to port the game from desktop version to mobile version.

During 2000 and 2010, cloud computing technologies were developed rapidly. Successful products such as Google Docs [2], Amazon Web service [3], Dropbox [4] gradually came into public view during that period. With the same concept as cloud computing [5], cloud gaming technology was also developed. The key idea of cloud gaming is to launch the game on the cloud servers, the then the video view will be captured and encode as video to transmit through network and reach user's device. On the reverse side, inputs and commands will be captured and transmitted back to the cloud server to execute the game logic. In this way, the end device does not need to have powerful CPU and GPU, not need to have large storage space and the game can theoretically be played on any operating system including operating system for mobile devices. However, one disadvantage is under constrained network conditions, there is a tradeoff here between quality and latency. Because the video encoding is a lossy process, if the video is encoded as very high quality, the bit rate increases and will thus introduce more latency. Otherwise, if the video is encoded to consume low bit rate, the latency will be low but the quality compromises. Some video streaming adaptation techniques [41] [42] were then developed to overcome this difficulty. It improves user experience to some extent but the problem can not be fully solved [6]. In 2010, the first commercial cloud gaming platform [23] was launched that is a sign showing cloud gaming architecture tended to be mature.

Since 2011, three technology trends have also been evolving rapidly, which makes extension of the basic cloud gaming architecture possible. First, the idea of virtual reality (VR) [7] and augmented reality (AR) [8] have been proposed again and

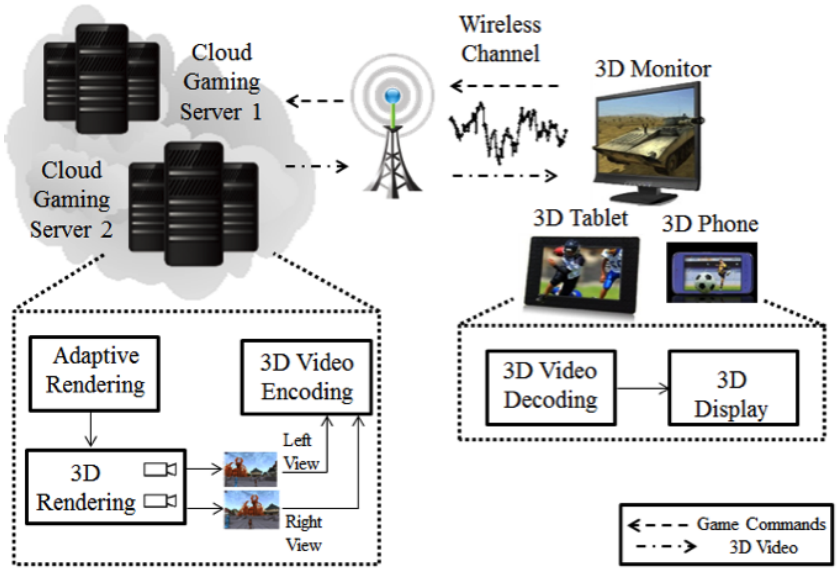


Figure 1.1. System architecture diagram of CMVIA(3D)

the industry has started developing virtual immersive applications on different devices and platforms. SecondLife [76] has developed virtual world application for desktop PCs. Oculus [9], HTC [10] and Sony [11] developed Head Mounted Devices (HMD) for consuming VR contents. Pokemon Go [12] became extremely popular from its first release day as an AR application running on mobile devices. Thus, we conclude that the cloud gaming architecture should not be constrained to work with games only, but can also extend to other virtual immersive applications. Second, since late 2010, mobile network infrastructure has started to be upgraded to support 4G-LTE standard [13]. It increases network bandwidth, decreases network packet loss and it also makes the data price to be lower for users. These changes made it possible to watch streamed videos on mobile devices and the same assumption applies to making cloud gaming to mobile device possible. Third, since 2011, mobile auto-stereoscopic 3D displays [14] have become more and more popular. Products based on new auto-stereoscopic 3D technology started entering the market. With huge success of 3D content at cinemas, it is believed that consumers would like their mobile devices equipped with 3D displays in the future.

Therefore, combining the above three technology trends, we propose an extension of the basic cloud gaming architecture called cloud mobile 3D display virtual immersive application CMVIA(3D) architecture. The system architecture diagram is shown in Figure 1.1. In detail, we place two virtual cameras in the virtual world to generate a left view and a right view of the virtual scene. After the two views are generated, they will be encoded as 3D video and transmitted through wireless network to the mobile device, and displayed on the device 3D screen. On the reverse side, control commands are captured and transmitted from mobile device to the server to execute. There exists an adaptive rendering module before 3D rendering engine. It is used to decide and set the best rendering parameters to ensure best user experience. The details of how this adaptive rendering module works will be explained in the following chapters.

1.2 Related Work and Existing Systems

In the following, we will first introduce some commercial cloud gaming systems from which we can see there still exists research challenges. Then related research areas will be carefully explored and investigated.

1.2.1 Commercial Cloud Gaming Platforms

In 2010, OnLive launched world's first commercial cloud gaming service platform [23]. It's monthly active users reached 1 million shortly after its establishment. In November 2010, powered by G-cluster technology, SFR launched a commercial cloud gaming service in France for IPTV users [24]. In 2011, Gaikai collaborated with game publishers to use cloud gaming technology to provide game trial experience to users [25]. It was acquired by Sony in 2012 [26]. In 2013, the first open-source cloud gaming platform GamingAnywhere was released [27]. In 2015, Sony Computer Entertainment acquired OnLive's patents and OnLive closed its doors [28]. Besides

commercial reasons, the technology provided by OnLive can only work under wired and WiFi networks but not under cellular networks. In October 2015, NVidia released its cloud gaming solution called GeForce Now together with its specialized hardware NVIDIA GRID [29]. The specification for GeForce Now indicated that a user must have over 10Mbps bandwidth with less that 100ms delay in order to launch the game. To conclude, all current commercial services support only wired and WiFi networks because of the challenges imposed by cellular wireless networks.

1.2.2 Video coding technology

For 3D virtual immersive applications, the main difference between traditional standalone rendering architecture versus cloud based rendering architecture is that the latter uses video encoding technology to compress the screen content as a video stream and then transmit. Therefore, using the latest video coding technologies and standards, reviewed below briefly, is the easiest way to reduce bitrate consumption and therefore reduce delay and improve user experience.

Video coding standards have evolved primarily through the development by ITU-T and ISO/IEC standards. H.261 [**?**] and H.263 [**?**] were produced by ITU-T while MPEG-1 [**?**] and MPEG-4 Visual [**?**] were produced by ISO/IEC. The two organizations jointly produced the H.262/MPEG-2 Video [**?**], H.264/MPEG-4 Advanced Video Coding (AVC) [**?**] and H.265/MEPG-H High Efficiency Video Coding (HEVC) [**?**] standards. The latest released standard HEVC is claimed to be able to encode a video using half of the bitrate that AVC does while producing the same quality. There are a lot of extensions of HEVC that have been developed and released afterwards among which there is an extension called Screen Content Coding [**?**]. SCC is designed to improve compression capability for video containing a significant portion of rendered (moving or static) graphics, text, or animation rather than camera-captured video scenes.

1.2.3 Stereo video streaming adaptation technology

In terms of video streaming, there exists many techniques that could adapt video bit rate according to network conditions to ensure low latency content delivery [38] [39]. Because stereo video or 3D video contains two views for user's left eye and right eye respectively, much more techniques were proposed to take advantage of this property and further improve the overall quality of service. Among them, one of the techniques allow left view and right view to be of asymmetric quality and let human brain to compensate for the degradation. Some experiments [40] [41] [42] show that as long as the view with lower quality is still better than some threshold, people won't notice it and would still generate the 3D view with acceptable quality in their brain. Thus, instead of tuning the video quality to be lower for both views when network bandwidth is not adequate, one of the views can preserve the quality while only the other is tuned to be lower and the final 3D quality would still be better than the former method with the same bit rate budget. The same concept inspired us to develop techniques in the rendering engine to do asymmetric rendering to further improve the use experience.

1.2.4 User experience model based on subjective tests

Vankeirsbilck et al. [43] proposed a user experience model for cloud gaming but it only considers video encoding impairment and it is based on a single view 2D display Cloud Mobile Gaming (CMG) system. Jarschel [**?**] and chen [45] conducted experiments to quantitatively measure quality of service for cloud gaming. Liu et al. [46] have done user experience study and modeling considering both video encoding impairment and graphics rendering impairment but it is also applicable only for 2D display CMG system. However, no user experience model for cloud mobile 3D display system has been proposed before.

1.2.5 Video bit rate model

Video bit rate modeling is a standard problem. Especially in traditional video encoding research area, bit rate needs to be accurately estimated so that rate control can be done correctly. For digital video broadcasting (DVB) transmission, strict CBR is widely used which makes rate control and bit rate modeling extremely important. Previous state-of-art methods usually use two pass methods to get statistical information in the first pass and decide the bit rate in the second pass [47]. Some other previous works model the relationship between video bitrate and encoding settings/video content features [48] [49]. However, there are several shortcomings. 1) These models are not specifically for screen content videos. 2) These models do not take into account graphics rendering settings. 3) These models are developed several years ago with videos of low resolutions. 4) Some models are developed based on old standards.

1.2.6 Adaptive rendering techniques

Hemmati et al. [50] propose the idea of selective rendering and has done some preliminary experiments to show that by not rendering some unimportant objects, video bitrate can be decreased. However, they applied their technique in 2D display CMG system and they did not answer the question about how to choose those objects, how it influences user experience and how to dynamically select the unimportant objects to be rendered according to network bandwidth. Wang et al. [51] take video encoding impairment, graphics rendering impairment and network impairment into consideration and propose an adaptation algorithm to minimize the overall impairment by choosing different levels of settings for different network conditions. However, the discrete number of levels restricts smooth transition between different settings.

1.3 Asymmetric Graphics Rendering

The key idea we introduce in this thesis is called asymmetric graphics rendering which is inspired by stereo video streaming adaptation in which two views for left eye and right eye can be of different objective qualities without compromising user perceived subjective quality. While normally it is done in the encoding process, we identify and demonstrate in this thesis that it can also been done in the rendering engine which is actually more flexible. Moreover, we notice and demonstrate that asymmetric graphics rendering can be used together with asymmetric encoding to further improve the user experience given limited bit rate budget. In this thesis, we start by applying asymmetric graphics rendering for the whole view and all objects in the view), and then gradually evolve the idea to apply it differently for each individual objects. Finally, we also consider to combine asymmetric graphics rendering with asymmetric encoding. Figure 1.2 illustrates the evolution of our proposed concepts of asymmetric graphics rendering.

We first briefly introduce some terms in rendering engine and encoder (more detailed introduction will be done in the next following chapters) and then explain the key ideas in Figure 1.2 one by one. First, in rendering engine, we define two parameters, called texture detail and view distance respectively. Texture detail defines the quality of texture map. It contains three levels, High, Medium and Low. Lower level causes more blurry effect to the view. View distance is defined as a distance threshold. When the distance from the camera to the object is larger than this threshold, rendering engine won't render it. Next, in video encoder, we define QP which means the quantization parameter. Because video encoding is a lossy process, quantization will create blurry artifacts to the original view. The larger the QP, the more blurry the view will be. We consider using the rendering parameters texture detail and view distance, and the encoding parameter QP, in this thesis. We can see in the first row of Figure 1.2, we enable to set asymmetric

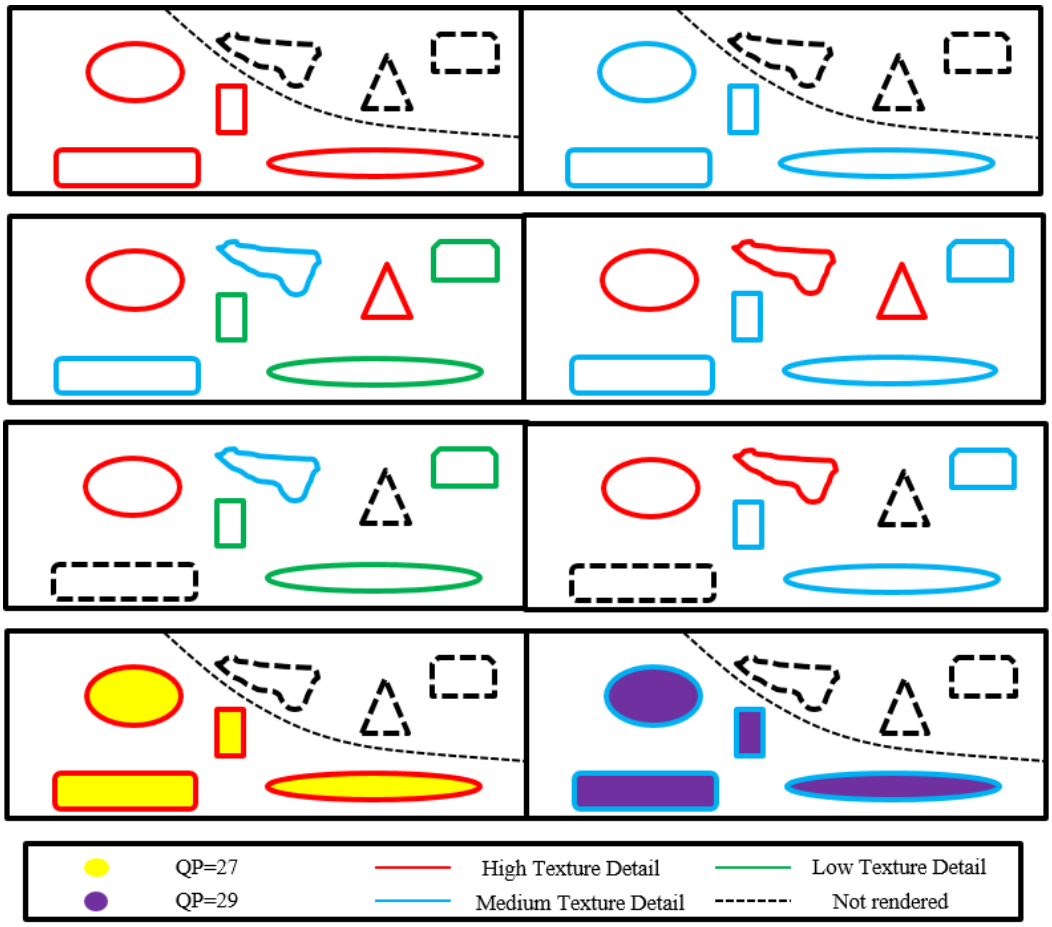


Figure 1.2. Evolution of asymmetric graphics rendering

texture detail and set symmetric view distance for two views. Both parameters can be adaptively tuned according to network conditions. In the second row, we consider setting asymmetric texture detail for each individual objects. The objects that attract more attention from users are regarded as more important and hence will be assigned to render with higher texture detail when possible. In the third row, we enhance the above idea by further allowing the object to be selectively rendered. Some objects with very low importance weight can choose not to be rendered at all. In the fourth row, we combine asymmetric graphics rendering and asymmetric encoding together and also allow symmetric view distance. These three parameters will be adaptively changed according to network bandwidth constraints.

1.4 Organization of the Thesis

In the following chapters, we describe in details the asymmetric rendering and encoding ideas introduced in Figure 1.2, and propose models, algorithms and architectures needed to implement the novel ideas. Specifically, Chapter 2 describes the asymmetric rendering technique described in the first row of Figure 1.2. Chapter 3 discusses the technique illustrated in the second and third rows of Figure 1.2, which is asymmetric and selective rendering for individual objects. Chapter 4 applies the idea of the last row of Figure 1.2 to further improve the cloud mobile 3D virtual immersive system. In each of the above chapters, the idea will be introduced in detail, subjective tests will be carried out to derive an user experience model, adaptation algorithms will be designed and final experiments will be conducted to verify the validity of the model and proposed techniques. Finally, Chapter 5 will conclude the whole thesis and point out possible future work in this area.

Chapter 2

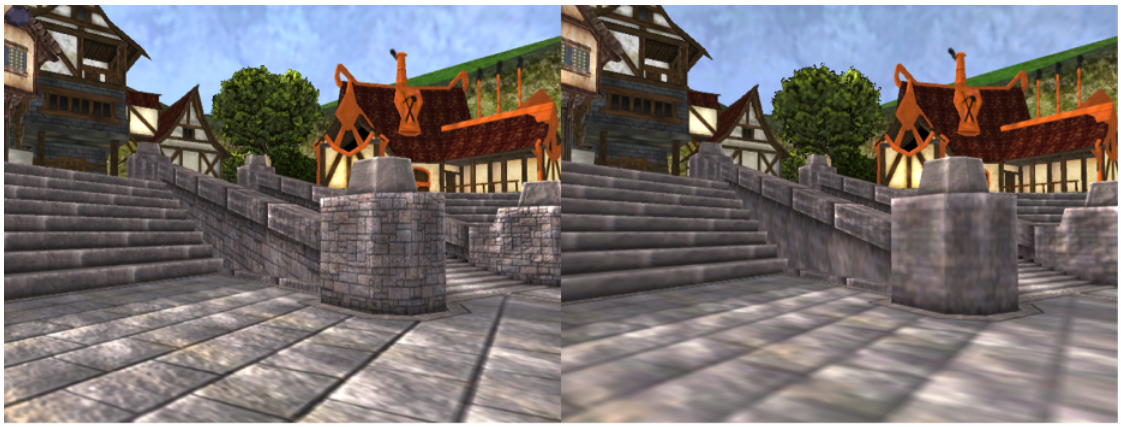
Asymmetric Graphics Rendering

2.1 Introduction

In this chapter, we study two rendering factors that have great impact on user experience and rendered video bitrate: texture detail and view distance. Texture detail defines the quality of the images on the surface of the objects. In detail, texture detail defines the quality of the images on the surface of the objects. We define texture detail to be high when the game is using the original texture images, to be medium when the texture images are downsampled once, and low when the texture images are downsampled twice. View distance determines which objects will be included in the rendered game video frame. For a given view distance value, except for the world boundaries (sky and ground), the game engine will only render the objects which are within the given distance from the camera to the object, and all the objects beyond this view distance value will be excluded. Since for 3D display games we generate two views using two virtual cameras (Figure 1.1), we can potentially render each view with different texture detail and view distance, leading to *asymmetric graphics rendering*.

Figure 2.1a shows an example of asymmetric view distance in which left view distance is set as 150 meters and right view distance is set as 100 meters. This means the objects which are more than 100 meters but less than 150 meters away from the camera

(a)



(b)

Figure 2.1. (a) top, Left view and right view with asymmetric view distance (b) bottom, Left view and right view with asymmetric texture detail

will be only rendered in left view but not in right one. Consequently, any object further than 100m (and less than 150m) can only be seen by the viewer's left eye, but not by the right eye. Figure 2.1b shows another example which we term asymmetric texture detail. In this example, left view texture detail is set as high quality and right one is medium quality. This means the surface quality experienced by the left eye will be slightly higher than that by the right eye.

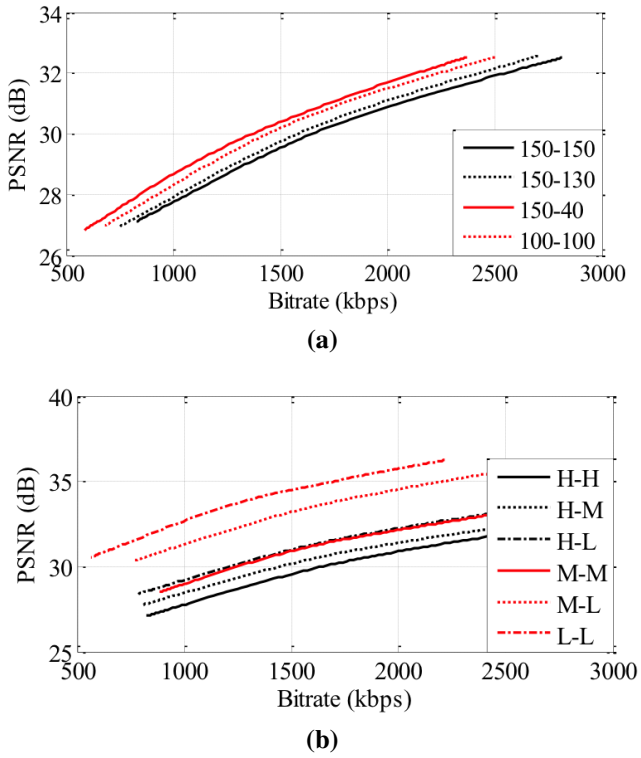


Figure 2.2. (a) top, Rate-distortion performance with different view distance setting for left view and right view (b) bottom, Rate-distortion performance with different texture detail settings for left view and right view

We will show in the following why asymmetric graphics rendering is promising to enhance user experience. Figure 2.2a and 2.2b present the rate-distortion comparison between different view distance settings and different texture detail settings respectively. (H stands for high texture detail, M for medium and L for low). We can see that by enabling asymmetric graphics rendering, video quality (as measured by PSNR) can

be potentially increased. For example, as shown in Figure 2.2b, the setting that has one view with medium texture detail and the other with low texture detail can have roughly 2dB PSNR gain over the setting where both the views have medium quality under the same encoding bitrate. In this way, user experience can be greatly improved either under the same network condition (increase video quality) or the same video quality (reduce bitrate needed and hence decrease network delay). However, while asymmetric graphic rendering can enhance the resulting video quality and/or reduce delay, it can also impair the surface quality; thereby impact the overall user experience. After introducing the concept of asymmetric rendering, in the following sections, we will derive impairment functions for the graphics rendering factors which can quantitatively measure how asymmetric graphics rendering will impact user experience.

2.2 Impairment Functions Derivation and Validation

In order to learn how graphics rendering factors affect user experience quantitatively, especially when asymmetric graphics rendering is applied, we define an impairment function I_R as Equation (2.1).

$$I_R = I_{VD} + I_{TD} \tag{2.1}$$

I_R indicates the impairment due to graphics rendering. It takes value between 0 and 100. Higher I_R value indicates larger impairment. Similarly, I_{VD} represents the impairment due to view distance, and I_{TD} indicates the impairment due to texture detail. In this section, we will derive impairment function I_{VD} and I_{TD} through subjective tests and then validate Equation (2.1) using another set of subjective tests.

In a typical 3D game, there are multiple different game scenes with different spatial characteristics; hence the rendering impairment caused by view distance and

texture detail may be affected by the game scene also. For instance, for the same view distance value, the graphics rendering impairment for an outdoor game scene where lots of objects are far from the camera may be very different from an indoor scene where lots of objects are close to the camera. To ensure that the impairment function for view distance, I_{VD}, is generally applicable to different game scenes, instead of modeling it as a function of the actual view distance value (in meters), we proposed to model it as a function of the percentage of missing objects caused by reducing view distance in [46]. In this work, we extended it to apply to asymmetric graphics rendering which enables different percentage of missing objects in different views. On the other hand, for I_{TD}, we will show by subjective test results that the impairment value will not be affected much by scene characteristics. In both cases, we will validate the derived impairment functions by including different game scenes in the subjective experiments.

2.2.1 Subjective Experiment Setting

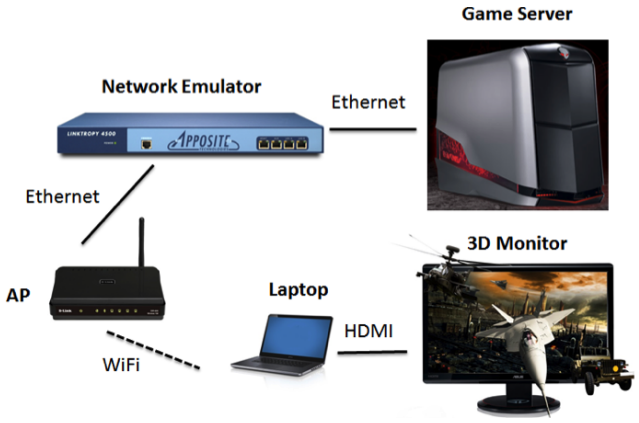


Figure 2.3. Testbed for subjective experiments

Figure 2.3 shows the testbed used for the subjective tests. We use a 3D monitor with a laptop to substitute for 3D display of mobile devices because current available mobile 3D displays do not have as good quality as 3D monitors which may cause additional impairment which we want to avoid. The laptop is connected to a network

emulator via an Access Point and the network emulator is connected to the game server. By changing parameters such as bandwidth, delay and packet loss on the network emulator, we can generate different kinds of network conditions. The selected game which runs upon the above framework is an online open source MMORPG game: PlaneShift [57]. To investigate how asymmetric graphics rendering factors affect user experience, we set the video encoding parameter QP to 25 which leads to sufficiently high encoding video quality and set network bandwidth to be sufficiently large so that only graphics rendering factors can cause impairment. We then invited 16 UCSD students (12 male, 4 female; aged 18$\sim$26) to participate in our subjective experiments. Firstly, we asked the testers to sit before a 23 inch LG D2342 3D Monitor, and showed them a 3D video as a training sequence before the real test started to let the testers adjust their viewing angle and viewing distance. The best viewing angle is related to the height of the testers but the best viewing distance is about 1 meter for all testers. After that, we start the game and manually set the graphics rendering factors according to Table 2.1 independently for each view. Once a combination of rendering factors is set, we ask the testers to play the game for 1 minute and evaluate the graphics rendering impairment according to the criterion listed in Table 2.2 at the end of each condition. During the whole experiment, the testers were asked to control the avatar to perform multiple tasks (including attacking an enemy, looking for an object, running, walking and talking to an NPC, etc.) under various different game scenes (including outdoor scenes like forest and city, as well as indoor scenes such as weapon store). In this experiment, a total of 51 combinations of rendering factors are studied. The experiment process including the test criterion and the number of participants adhere to the ITU-T Recommendations [58, 66]. We collect all the evaluations under different graphics rendering factors and use them to derive the impairment functions.

Note that in Table 2.1, we have used the percentage of missing objects instead

of view distance (in meters) in order to make the I_{VD} function applicable for different scenes. We can easily compute view distance from percentage of missing objects using the following method: suppose we need to set the percentage of missing objects to be 10%, we firstly extract the distance information of all the objects (this information can be easily fetched during the rendering process), then we choose the 90^{th} percentile of all the distance values to be the corresponding view distance (for 10% missing object). This is because the 90^{th} percentile can ensure that 90% of objects have less distance than it, therefore there are 10% of objects that exceed the view distance and will be missing.

Table 2.1. Graphics rendering factor setting

Factors	**Experiment Values**
Texture Detail(Down Sample)	High(0) Medium(2) Low(4)
Percentace of Missing Objects (%)	100 90 80 70 60 50 40 30 20 10 0

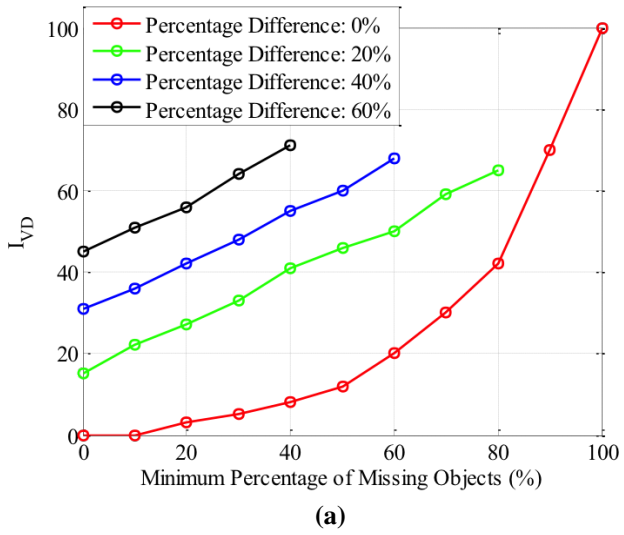
Table 2.2. 3D graphics quality and criterion for I_R

I_R	**Description**
0	Excellent depth perception, Excellent visual quality, no visual impairment
0-20	Good depth perception, Good visual quality, minor visual impairment, will continue the game
20-40	Acceptable depth perception but noticeable visual impairment, might quit the game
40-60	Can still get feeling of depth perception but clear visual impairment, usually quit the game
60-100	No feeling of depth at all, unacceptable quality, definitely quit the game

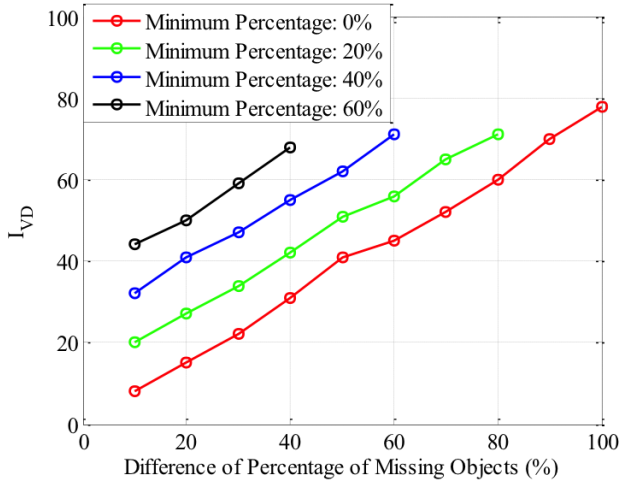
2.2.2 Impairment Function Derivation

To derive I_{VD} and I_{TD}, we use the results from the experiments where we only vary one rendering factor shown in Table 2.1 and keep the other at its best value. For example, to derive I_{VD}, we set different combinations of the percentage of missing objects for the left and right views according to Table 2.1, while keeping the texture details of

the two views to be both at the highest value, such that all the impairment is caused by view distance.



(a)



(b)

Figure 2.4. Subjective test results: (a) top, Relationship between impairment values and minimum percentage of missing objects (b) bottom, Relationship between impairment values and difference of percentage of missing objects

In the following, we first derive the impairment due to view distance, I_{VD} which mainly depends on two factors: (a) the value of the minimum percentage of missing objects among the two views, and (b) the difference of percentage of missing objects

between the two views. The minimum percentage of missing objects indicates how many objects are missing in both views and the value of difference represents how many objects are seen by one eye but not the other. Figure 2.4a shows the relationship between I_{VD} and minimum percentage of missing objects while Figure 2.4b shows the relationship between I_{VD} and difference of percentage of missing objects. From these two figures, we can observe that there is a very different relationship between I_{VD} and minimum percentage of missing objects whether the two views have the same percentage of missing objects or different. When the two views have the same percentage of missing objects which means the difference percentage is 0%, we can see I_{VD} has a non-linear relationship with minimum percentage of missing objects. Thus, we use nonlinear regression method to derive the equation of I_{VD} in this case. We tried to use square, third power and fourth power to do the regression. The regression square errors for the three methods are 301.09, 30.58 and 15.74 respectively. Since the error of square is big, we do not consider it. However, the error difference between third power and fourth power is not large. In addition, using lower power can avoid over fitting and potentially save computational power for the optimization algorithm we would propose later. Therefore, we propose to use Equation (2.2) to describe the relationship when both views have the same percentage of missing objects.

$$I_{VD} = aP^3 + bP^2 + cP \tag{2.2}$$

in which P denotes the percentage of missing objects and a, b, c, d are four parameters. For the game Planeshift, $a = 0.000179, b = -0.0115, c = 0.3497$. Note that although the above impairment model as well as all the models derived subsequently is general, the values of the parameters need to be derived for specific games. We will validate the above statement by applying the models to another game of a very different genre in the next section.

For the other cases when the difference of percentage of missing objects is not zero, we observe that the impairment has a bilinear relationship with the two factors. Thus, we use a bilinear regression method to derive the equation of I_{VD}. Equation (2.3) shows the relationship. P_{diff} means the difference of percentage of missing objects and $P_{\min}$ means the minimum percentage of missing objects.

$$I_{VD} = eP_{diff} + fP_{\min} + g \tag{2.3}$$

in which

$$P_{diff} = |P_1 - P_2|$$

$$P_{\min} = \min(P_1, P_2)$$

and $e = 0.754, f = 0.608, g = 0.98$ for Planeshift.

A similar method is used to derive the impairment function of texture detail, I_{TD}, where we vary the texture detail setting of left and right views, change different scenes, and collect participants' scores according to Table 2.2. The testers were asked to give two kinds of scores, combined score for a combination of different scenes and individual score for a specific scene. Table 2.3 summarizes the results. From Table 2.3 we can make the following observation. The difference between individual scores and combined score ranges between -9.7% and 9.2% and thus we believe the complexity of the scene does not influence the score too much, instead, the texture detail dominates the impairment. In addition, the additional impairment due to asymmetric texture detail of left and right views is marginal (only about 7.5) when the asymmetry is of one level (like High-Medium, or Medium-Low); however, the impairment can go up significantly (more than 25) if the asymmetry is higher (like High-Low). Based on the above observations, we denote high texture detail as a value of 0, medium as 1 and low as 2 and further derive

the formula for I_{TD} (as shown in Equation (2.4)) using regression method. Note that in Equation (2.4) we use a square term, TD_{diff}^2, to penalize the large impairment due to high texture asymmetry of two views (like High-Low). Term TD_{max} is used to indicate the impairment due to low surface quality itself.

$$I_{TD} = hTD_{diff}{}^2 + iTD_{diff} + jTD_{\max} \tag{2.4}$$

in which

$$TD_{diff} = |TD_1 - TD_2|$$

$$TD_{\max} = \max(TD_1, TD_2)$$

and $h = 6.6591, i = 0.4318, j = 12.1932$ for Planeshift.

2.2.3 Impairment Function Verification

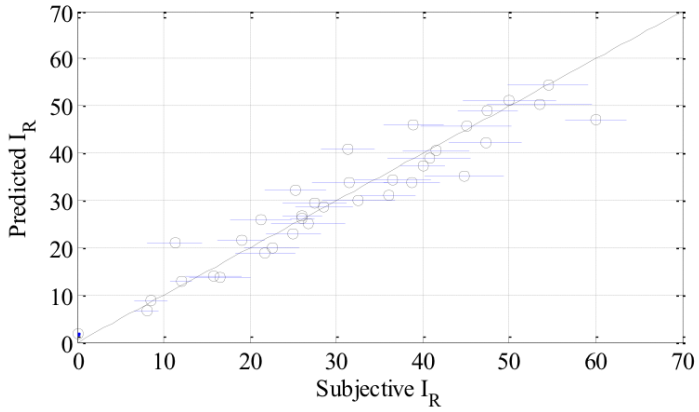


Figure 2.5. Validation of I_R with blue line showing 95% confidence interval

In order to verify the functions (Equations (2.1),(2.2),(2.3),(2.4) and Table 2.3) derived in the previous subsection, we conducted another set of experiments with a new group of 15 participants (11 male, 4 female; aged 18∼25), playing the same game, and evaluate using the same criterion (Table 2.2); however, in this set of experiments, the

texture detail and percentage of missing objects parameters are changed at the same time. Figure 2.5 shows the relationship between predicted I_R computed by the derived impairment function (y-axis) and subjective I_R given by human subjects (x-axis). We also plotted 95% confidence interval for each measurement as blue lines in the figure. The correlation is 0.97, which proves the accuracy of the derived impairment functions.

Note that one limitation of our subjective study is that when we asked the testers to evaluate the graphics rendering impairment while playing the game, they were asked to play for 1 minute and give the score. The 1 minute playing time is a short period and the subjective score may not capture the visual discomfort caused by assigning different view distances to the two views, such that the player will see some objects visible in one view but not visible in the other view. This discomfort effect of asymmetric rendering will be further studied in our future work by asking the testers to play the game for longer time and evaluate their experience.

Table 2.3. Subjective test results: average I_{TD} scores for different texture detail combinations and combined/individual scenes

Texture Detail Combination	H-H	H-M	H-L	M-M	M-L	L-L
Combined Score I_{TD}	0	7.625	27.5	11.5	18.75	25
Forest Scene Score I_{TD}	0	7.30	27.85	10.38	19.84	25.38
Plaza Scene Score I_{TD}	0	8.30	28.85	11.61	20.46	24.00
Lane Scene Score I_{TD}	0	7.84	25.61	12.30	20.76	23.07

Table 2.4. Settings for network delay factor

Factor	**Experiment Values**
Network delay (ms)	80 120 160 200 250 300 350 380 400 450 480 500 550 580 600 650 680 700 750 780 800 1000

2.3 Overall User Experience Modeling

In [46, 52–54, 59], the factors affecting Cloud Mobile Gaming User Experience have been analyzed, and a UE Model has been proposed which takes into account the impairments due to these factors. Though the above applies to 2D video streamed and displayed on 2D mobile devices, the category of factors are the same in the 3D case. As is described in [46], there are three major categories of objective factors: graphics rendering factors, video encoding factors, and mobile network factors. As discussed in the previous section, graphics rendering factors include texture detail and view distance which affects the user perceived surface quality of the graphics. Video encoding factors include video quality, video bitrate, etc. which affect both visual quality and response time. Mobile network factors include packet loss rate and network delay which will affect visual quality and response time. In this work, we will focus on studying graphics rendering factors and mobile network factors, while keeping the video encoding factors at high level such that there will be no impairment caused by video encoding. As for graphics rendering factors, impairment functions have already been derived in the previous section. As for mobile network factors, we will only consider network delay, since our use of TCP minimizes any impairment due to packet loss while increasing the potential of impairment due to delay. We also show in the next subsection that we can reuse the same impairment function for network delay as has been derived for the 2D case [52]. After getting impairment functions separately for graphics rendering factors and network delay factors, we will derive and validate an overall user experience model for CMG(3D).

2.3.1 Impairment Function Validation for Network Delay

In our previous work [52], the impairment caused by network delay for 2D cloud gaming has been studied and the impairment function is as follows:

$$I_D = \begin{cases} 0 & (T_1 > Delay > 0) \\ T_0\left[(Delay - T_1)/(T_2 - T_1)\right] & (T_2 > Delay > T_1) \\ T_0 + \alpha\,(Delay - T_2) & (Delay > T_2) \end{cases} \tag{2.5}$$

For the game Planeshift, $T_0 = 40, T_1 = 120, T_2 = 440, \alpha = 0.05$.

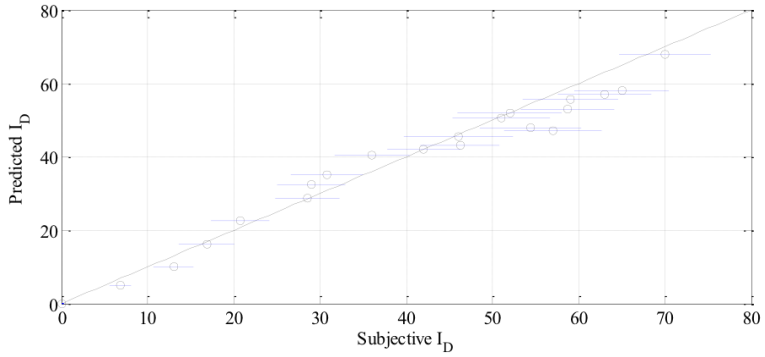


Figure 2.6. Relationship between predicted and subjective I_D value with blue line showing 95% confidence interval

We validate the accuracy of this function for 3D cloud mobile gaming by conducting subjective tests with a new group of 15 participants (11 male, 4 female; aged 18$\sim$25) playing the game Planeshift using our CMG (3D) testbed shown in Figure 2.3. The value of network delay parameters are shown in Table 2.4. The tasks which the testers are asked to perform; the scene they passed and the duration of the time they play are the same as the test for derivation. Figure 2.6 shows the relationship between subjective impairment given by people with the objective impairment computed using Equation (2.5). We also plotted 95% confidence interval for each measurement as blue lines in the figure. The correlation between predicted ID (y-axis) and subjective ID (x-axis) is 0.97. This high

correlation indicates that although Equation (2.5) is derived for 2D cloud gaming, we can also apply it in 3D cloud gaming. We next derive the UE model based on these two impairment functions.

2.3.2 CMG(3D)-UE Model Derivation and Validation

In this subsection, we propose a Cloud Mobile 3D Display Gaming User Experience (CMG(3D)-UE) model by taking network delay impairment and asymmetric graphics rendering impairment into consideration. Derivation and validation are both done through subjective tests.

We define Cloud Mobile 3D Display Gaming Mean Opinion Score (CMG(3D)-MOS) as a measurement metric for CMG(3D)-UE. In order to formulate it using the impairment functions, we follow the same framework as ITU-T E-model [58]. The ITU-T model offers a way to quantify the combined impact of several audio transmission impairments, including network delay impairment, audio distortion impairment, etc. Hence, although the model is originally developed for audio transmission, since the impairment for cloud mobile gaming also includes combined effects of rendering impairment and network delay impairment, we borrow the framework of the ITU-T model for our study. The function of CMG(3D)-MOS formulated by R factor can be found in ITU-T E-model. We duplicate it for our CMG(3D)-MOS formulation:

$$\text{CMG(3D)-MOS} = 1 + 0.035R + 7 \times 10^{-6} R\,(R-60)\,(100-R) \tag{2.6}$$

In Equation (2.6) the transmission rating factor, R, takes value from range $[0, 100]$ (the higher R, the better CMG(3D)-MOS). CMG(3D)-MOS relates to R through a non-linear mapping and takes value in the range $[1, 4.5]$. Considering that R in ITU-T E-Model is designed specifically for audio transmission application and not suitable for CMG

applications where graphic rendering factors affect user experience, we propose the following new definition of R:

$$R = 100 - I = 100 - I_R - I_D + C_{DR}(\sqrt{I_D \cdot I_R}) \tag{2.7}$$

in which I means the overall impairment, I_R is the impairment due to graphics rendering factors and I_D is the impairment due to network delay factor. The last term adjusts I by the cross effect between the two impairments, and C_{DR} is a constant.

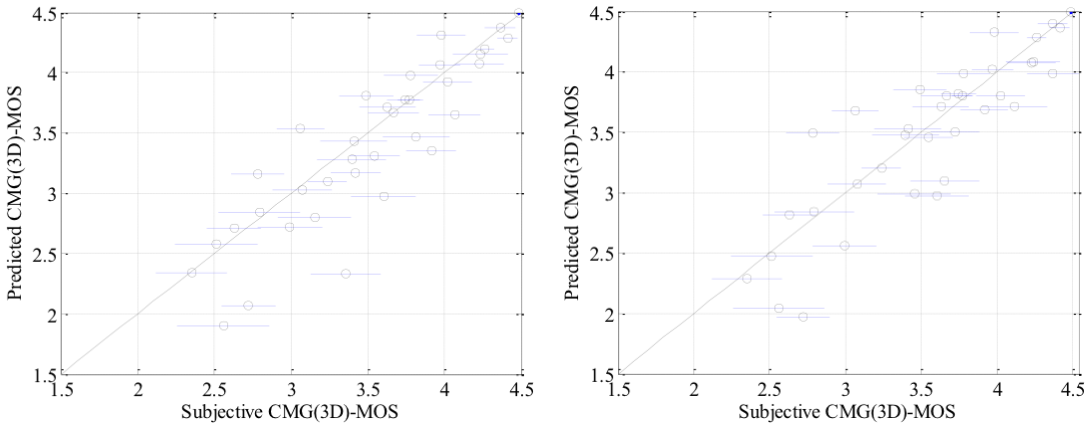


Figure 2.7. (a) left, Relationship between predicted MOS and subjective MOS for the game Planeshift, (b) right, Relationship between predicted MOS and subjective MOS for the game Broadsides, Blue line showing 95% confidence interval

In order to derive the parameter C_{DR} and verify the correctness of the model, we conduct a new series of subjective experiments using the same subjects used earlier to validate the impairment functions. This time we simultaneously change the graphics rendering factors at the cloud game server and the network delay factor at the network emulator, and let the testers give their scores according to Table 2.5 whose scale is mapped to Table 2.2 via Equation (2.6). We use 60% of the data to derive C_{DR} and use the rest 40% to validate. Our results show that $C_{DR} = 0.4$ for Planeshift. Then, we use

Equations (2.6) and (2.7) to compute our predicted UE score and compare it with the subjective UE score given by the testers to validate the model. Figure 2.7a shows our results. We also plotted 95% confidence interval for each measurement as blue lines in the figure. We can see from the figure that the correlation between predicted UE score and subjective UE score for the game Planeshift is 0.93, indicating adequate accuracy of our CMG(3D)-UE model.

To demonstrate the applicability of the CMG(3D)-UE model to other games, we applied it to another 3D game Broadsides [63], which belongs to a different genre than game PlaneShift. For the new game, we apply the same approach for training and validating the CMG(3D)-UE model. We conduct a new group of subjective tests of 16 participants (10 male, 6 female; aged 19$\sim$28) and collect evaluations under different combinations of the rendering and network factors. Firstly we use the results where only one factor varies and all the others are fixed at the best values to train the impairment functions for the new game. Table 2.6 shows the coefficient values. Comparing these values with the coefficients for game PlaneShift, we can see that for different games, the coefficients are different. Secondly we use the results where all the factors vary simultaneously to derive and validate the overall CMG(3D)-UE model. 60% of the results are used for training the model of R and computing the coefficient C_{DR}, and the other 40% of test results are used for validating the model. Figure 2.7b is the scatter plot of the relationship between subjective and predicted CMG(3D)-MOS. For the new game Broadsides, $C_{DR} = 0.2$ and the correlation is 0.87. From Table 2.6 and Figure 2.7b, we can see that for the new game, after training and refitting the coefficients, the proposed CMG(3D)-UE model will also lead to high modeling accuracy.

Table 2.5. 3D graphics quality and criterion for CMG (3D)

CMG (3D)-MOS	**Description**
4.5	Excellent depth perception, Excellent visual quality, no visual impairment and excellent response time, no noticeable delay.
4.0-4.5	Good depth perception, Good visual quality, minor visual impairment and good response time, not easily noticeable delay, will continue the game
3.0-4.0	Acceptable depth perception but noticeable visual impairment, or noticeable delay but can endure, might quit the game
2.0-3.0	Can still get feeling of depth perception but clearly visual impairment, can still control the game, but have to wait for the response, usually quit the game
1.0-2.0	No feeling of depth at all, unacceptable quality, or unacceptable delay definitely quit the game

Table 2.6. Parameters for game broadsides

a	b	c	e	f	g	h
0.000234	-0.0089	0.435	0.863	0.547	2.08	6.53
i	j	T_0	T_1	T_2	α	
0.45	11.98	40	100	400	0.045	

2.4 Asymmetric Rendering Adaptation Approach

In Cloud Mobile 3D Display Gaming, game video has to be transmitted through wireless network, the latter characterized by unpredictable bandwidth fluctuations. An effective approach to cope with the bandwidth fluctuation of mobile network is to adapt the graphics rendering factors such that the required encoding bitrate can be adapted and maintained below the available bandwidth and therefore avoid network congestion and hence delay. However, as illustrated in Figure 2.2, while lowering the rendering factor is effective in reducing network delay, it can cause negative impact on graphics quality, and hence will influence the overall user experience. In other words, under a given network bandwidth budget, there is a tradeoff between network delay and graphics rendering quality. The question that needs to be addressed is how to select the optimal rendering factors such that the overall user experience is maximized, when taking into account both graphics rendering factors and network delay factor. Previous work [46, 53] have proposed techniques to dynamically change graphics rendering factors according to network conditions, but since those solutions were for streaming 2D rendered video, they did not consider using asymmetric graphics rendering. Thus, in this section, we use the proposed CMG(3D)-UE model to derive an optimization algorithm which can take asymmetric graphics rendering into consideration to solve this problem. The aim of our optimization algorithm is to maximize CMG(3D)-MOS which is equivalent to minimizing I. We also provide bounds for both network delay factor and graphics rendering factors to ensure that the solution for the problem will not let one aspect of the user experience to be too good while letting the other aspect to be too bad. Thus, we formulate the asymmetric graphics rendering optimization problem as is shown in Figure 2.8:

It may be most prudent to specify the bounds most appropriate to a specific

Given:
Network bandwidth BW
Find:
The optimal graphics rendering factors, including P_L, P_R, TD_L and TD_R, to minimize I

$$I^{opt} = \min I = I_R + I_D - C_{DR}(\sqrt{I_D \times I_R}) \tag{2.8}$$

s.t.

$$P_{lower\_bound} \leq P_R \leq P_L \leq P_{upper\_bound}$$
$$TD_{lower\_bound} \leq TD_L \leq TD_R \leq TD_{upper\_bound}$$
$$Delay \leq D_{Threshold}$$

Figure 2.8. Problem formulation

game. For example, the following values are appropriate for Planeshift: $P_{lower\_bound} = 0$, $P_{upper\_bound} = 80$, $TD_{lower\_bound} = 0$, $TD_{upper\_bound} = 2$, $D_{Threshold} = 120$ms. For a social game, a more relaxed value of $D_{Threshold}$ may be sufficient.

Note that our subjective experiments show that exchanging rendering settings between left and right views result in almost the same user experience scores, thus in order to tighten the variable range and hence decrease complexity, in the above problem formulation, we let the left view always have better or equal graphics rendering quality compared to the right view.

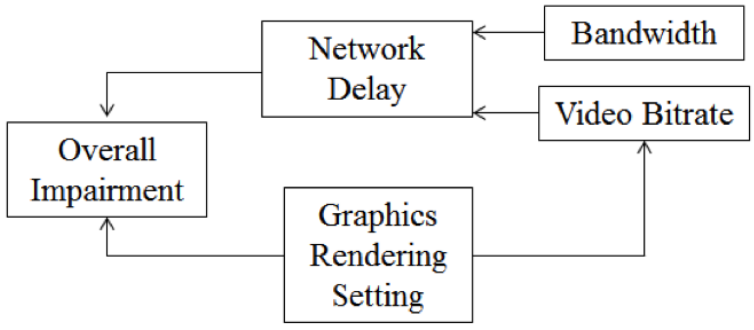


Figure 2.9. Factors affecting overall impairments

Next, we propose an approach to solve the above optimization problem. Considering that during the game session, the network dynamically changes. Hence, our approach

is to divide time into constant intervals, and apply the proposed solution periodically so that the overall impairment I during every time interval is minimized. We will dynamically select the optimal graphics rendering factors for the left and right views to adjust to the dynamic fluctuations in the network. As shown in Equation (2.8) and Figure 2.9, the overall impairment I is determined by the selected graphics rendering factors and the expected delay. The expected delay mainly depends on two factors: the available mobile network bandwidth of the next interval, and the amount of video data generated during the next time interval. Regarding these two factors, we assume the network bandwidth can be estimated using a probe-packet approach which will be discussed in Subsection 2.4.2. And the amount of generated video data is solely dependent on graphics rendering factors. Therefore, the proposed optimization problem can be restated such that, both the optimization target (the overall impairment I) and the constraint function (the experienced delay) will be solely dependent on the graphics rendering factors. We use a branch and bound based algorithm to solve this problem which will be discussed in Subsection 2.4.3.

The rest of this section is organized as follows. In Subsection 2.4.1, we propose and verify a model to predict the bitrate of the encoded 3D video resulting from rendering performed with given rendering factors. Subsection 2.4.2 proposes and verifies an approach to predict network delay given the available network bandwidth and bitrate of the 3D video that needs to be streamed from the cloud. Subsection 2.4.3 proposes the overall algorithm to solve the proposed optimization problem.

2.4.1 Relationship between Graphics Rendering Factors and 3D Video Bitrate

In our proposed CMG(3D) platform, the left and right views are firstly rendered and then encoded. Our approach is to change rendering factors to impact the video bitrate, while ensuring high video quality (no video encoding impairment) by using high

(constant) QP during video encoding. Thus, we need to estimate the effect of changing the rendering factors on resulting video bitrate. However, using constant QP can result in big fluctuations in video bitrate, making estimating resulting bitrate from rendering factors used very difficult. In this subsection, we first explain the two factors which can influence video bitrate when using constant QP, and how we can minimize their influence so the bitrate will be primarily influenced by the graphics rendering factors. We next derive a model to estimate video bitrate of both the views as a function of the rendering factors used to render the videos, and provide results validating the accuracy of the estimation model.

As is commonly known, encoding with constant QP may cause a wide range of bitrate changes, especially between static scenes and high motion scenes. That is because static scenes will contain a number of blocks which are encoded as SKIP mode and they will cause much fewer bits than any other modes in H.264 standard. In addition, the GOP setting also influences the bitrate much because GOP defines how frequently an I frame appears; in constant QP case, every I frame will cause much higher bitrate than P frame or B frame. In this way, if the GOP is not set properly, the bitrate may be high in one second but much lower in another. Next, we describe how we minimize the uncertainty in the video bitrate due to the above two factors that are motion and GOP size.

Firstly, we empirically observe that game videos are mostly high motion. Our assumption is verified by a study [60] which shows 3D game has really high motion energy (the average difference between consecutive frames). However, there can be cases of mostly static scenes, like during temporary pauses by the gamer. To eliminate the effect of static scenes on the resulting bitrate, we stop encoding and streaming the game video when we detect static scenes. Consequently, the 3D game video streamed from our system will be mostly very high motion, eliminating the uncertainty on bitrate due to different types of motion.

Secondly, in order to eliminate the influence of GOP size and the burst bitrate consumed by I frames, we take use of "intra refresh" technique [61] in our system to balance bitrate. This technique does not require any I frame in the whole video except the first frame, but instead, it requires every frame to have a portion of blocks which is encoded by intra modes. In this way, for example, if the GOP size is 30 then every frame will have 1/30 blocks which are not overlapped in position with each other to be encoded by intra modes. We implement the above in x264 encoder using the "–intra-refresh" option. Having minimized the influence of motion and I frame/GOP size on resulting bitrate, for a given resolution and frame rate, the bitrate for every frame mainly depends on the content of video frame, which in our case is influenced by the graphics rendering factors. Furthermore, because we use H.264 standard to encode two views separately, the total bitrate consists of the bitrate for the left view plus the bitrate for the right view.

$$B_{SUM} = B_L + B_R \tag{2.9}$$

Note that the relationship between graphics rendering factors and video bitrate are the same for the left view and the right view, so in the following we derive this relationship based on a single view. Using our CMG-(3D) prototype, we collect data by capturing 1 hours' game video data. The data contains different players performing different tasks in different scenes with different graphics rendering settings. We use 40 minutes' data to derive and 20 minutes' data to verify. The resolution of the video is 640×480, the framerate is 25fps and QP is set to be 25.

To derive the relationship, we calculate the average bitrate for each combination of the graphics rendering factors and plot it on Figure 2.10. From this figure, we can get two main observations. Firstly, for a given texture details level, the relationship between the VD and bitrate is almost linear. In this way, a linear term for view distance is used in

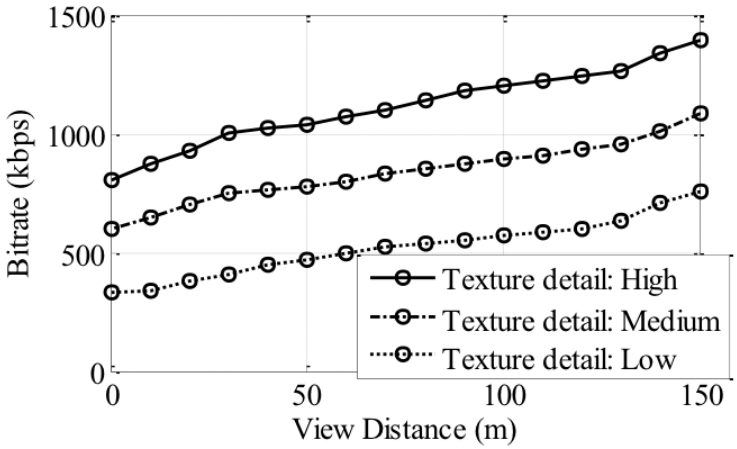


Figure 2.10. Derivation of relationship between graphics rendering factors and video bitrate

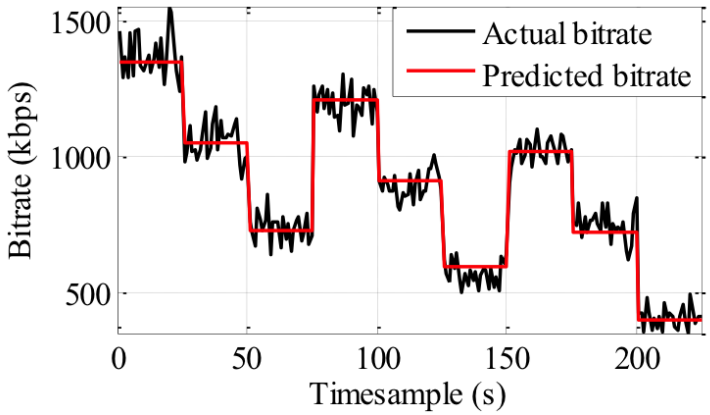


Figure 2.11. Verification of the relationship between graphic rendering factors and video bitrate

our derived function. Secondly, the relationship between the TD and bitrate is not linear which is because the image has two dimensions and the texture detail level relates to both dimensions so that we include a square term of texture detail in the derivation. According to the observations mentioned above, a regression method is used to derive the following relationship between graphics rendering factors and video bitrate.

$$\begin{aligned} B_L &= kTD_L{}^2 + lTD_L + mVD_L + n \\ B_R &= kTD_R{}^2 + lTD_R + mVD_R + n \end{aligned} \tag{2.10}$$

For the game Planeshift, we derive the values of the coefficients using the first 40 minutes data as: $k = -8.5, l = -291.5, m = 2.7, n = 937.4$.

Figure 2.11 shows the validation results. Predicted values (computed using Equation (2.11)) are represented by the red line while actual bitrate values are shown by black line. We can see that the predicted bitrate values are very close to the actual bitrate values. The mean error rate is 5.38% which proves the accuracy of our proposed model.

2.4.2 Delay Prediction and Verification

Previous work has been done to model the network and predict delay (like [64,65]). However, most of these techniques are based on transport layer or network layer, requiring information like TCP packet header. In contrast, we would like to develop an application layer delay prediction technique, which can be easily deployed by our application layer CMG(3D) approach. In the following, we show the derivation of the predicted delay model assuming that we know the available bandwidth and the video bitrate.

At any time t, the total experienced delay consist of three parts, server delay caused by graphics rendering and video encoder tasks, network delay due to bandwidth constraint, and propagation delay indicating the natural delay from a source to a destination depending on geographic distance and transmission medium. Equation (2.12) shows

this relationship.

$$D(t) = D_N(t) + D_S(t) + D_P(t) \tag{2.11}$$

in which $D_N(t)$ stands for network transmitting delay at time t, $D_S(t)$ represents server (computation) delay and $D_P(t)$ means propagation delay. In our work, we assume encoding delay and propagation delay are constant at any time. All the delay components are in units of second. Equation (2.13) and (2.14) show how $D_N(t)$ is calculated.

$$D_N(t) = \max\left(\left(S_P(t) + S_C(t)\right)/BW(t) - T_{INT}, 0\right) \tag{2.12}$$

$$S_C(t) = B(t) \cdot T_{INT} \tag{2.13}$$

$$S_P(t + T_{INT}) = \max\left(S_P(t) + S_C(t) - BW(t) \cdot T_{INT}, 0\right) \tag{2.14}$$

in which $BW(t)$ is the current bandwidth, $B(t)$ is the current video bitrate, $S_C(t)$ is current generated data size and $S_P(t)$ is data size of previous accumulated data in the streamer buffer. For every time interval T_{INT}, current generated data whose size is $S_C(t)$ will be queued in streamer buffer on the server which is equal to the product of video bitrate and time interval. However, the buffer may have some existing data, with size $S_P(t)$, from previous time period which has not been streamed yet. We will discuss later how $S_P(t)$ can be calculated. As shown in Equation (2.13), it will take $(S_P(t) + S_C(t))/BW(t)$ seconds for this new data to be streamed to the mobile device over the wireless network. If the required time is less than T_{INT}, it will not create delay $(D_N(t) = 0)$, but if it exceeds T_{INT}, the expected delay will be the excess time (required streaming time minus time interval T_{INT}). After the expected delay is calculated, we use Equation (2.14) to update the buffer data size remaining to be transmitted in the next time interval. It first adds the previous data and current data together and then subtracts it by the product of bandwidth and time interval. If the result is less than zero, it means there is no data remaining so

that we set $S_P(t + T_{INT})$ to be zero. Note that in our experimental setup, we use $T_{INT} = 1$ second.

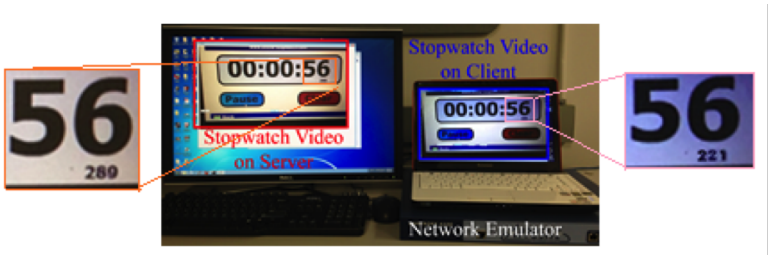


Figure 2.12. Testbed for measuring delay

We verify the accuracy of the above procedure by conducting the following experiment. Figure 2.12 shows our testbed setup. In the experiment, we stream a pre-recorded video from a server (a desktop computer) to a client (a laptop computer) through a network emulator. As the video is streamed, we apply a network bandwidth profile on the network emulator to control the bandwidth and generate network delay. The pre-recorded video is encoded with constant bitrate of 2Mbps and framerate of 60fps. Note that during the encoding process, the default CBR rate control has 10% inaccuracy, hence we have used the "–nal-hard" option in x264 encoder which will add dummy bits to achieve higher accuracy(within 0.1%) in rate control. The content of the video is a stopwatch that shows time going in the resolution of millisecond. We also programmed a pair of software (server and client) such that on the server side, the video will be displayed simultaneously as it is transmitted, and on the client side the video will be decoded and displayed once it is being received. Because the content of the video is a stopwatch and both the server and client are playing the video, the latency in this case can be easily calculated by subtracting the time shown on the server from the time showing on the client. We use a high resolution camera to record a video of the whole process and process the data offline.

Figure 2.13 shows the result. Black dotted line is the network bandwidth trace

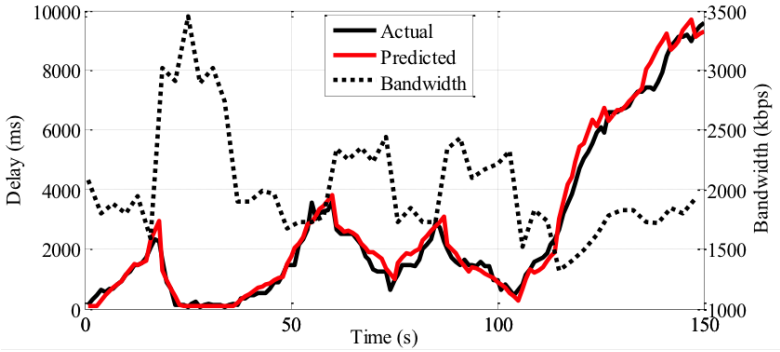


Figure 2.13. Results showing actual and predicted delay

captured in a real 4G-LTE environment. Red solid line shows the predicted delay results calculated by Equation (2.11)$\sim$ (2.13) and black solid line shows the actual experienced delay measured. With regard to the propagation delay, $D_p(t)$, and the server encoding delay, $D_S(t)$, in the theoretical calculation (Equation (2.12)), because the server and client is close enough, we set $D_P(t)$ to be 0 and also because the video is pre-encoded, so we also set $D_S(t)$ to be 0. Considering that the accuracy resolution of 60fps video is about 16ms, we claim from Figure 2.13 that the predicted results match the actual experimental results very well. Note that though in the above experiment we consider the case where $D_s(t)$ and $D_p(t)$ values are zero, in our CMG(3D) application they will not be. As we discuss later in the next section, they can still be estimated in advance, depending on the cloud service and servers used, and hence can be used to estimate $D(t)$ in Equation (2.12).

Thus, the above results show that if we know the video bitrate and network bandwidth, we will be able to predict the delay in an accurate manner. We next briefly describe how we estimate the network bandwidth. Various techniques have been proposed, like pathChirp [55] and BART [56] to accurately estimate the network bandwidth by injecting probing traffic. BART [56] uses fixed inter-packet separation probe packet train together with a Kalman filter based method to do real time bandwidth estimation

with demonstrated accuracy better than pathChirp. However, the additional probe traffic overhead of the above techniques can be very high; for example [56] requires 0.2 Mbps overhead to achieve reasonable accuracy. In our approach, we use BART [56] with the following modifications so that we can use the video data we anyway need to transmit instead of additional probing traffic, thus avoiding overhead.

The original BART algorithm sends a series of 1.5KB packets from server to client. By calculating the receiving time interval and the sending rate value injected in the packets, it will be able to estimate the current bandwidth. In our CMG(3D) platform, we generate video data at a certain framerate, transmitting one frame as several 1.5KB packets right after it is generated. In order to integrate BART algorithm into our platform, in other words, to let the client know the sending rate as well as to instruct the server to generate some predefined time interval between sending two 1.5KB packets, we did the following two main changes.

1) For all of the data packets of each frame, we set time interval between sending two packets of the same frame (originally there is no time interval between these two packets so that one packet is sent right after another), but we ensure that the total time interval to be less than 1/framerate sec, otherwise it will create additional delay for the next frame.

2) We insert the sending rate as a value into the video packet to send to the client as BART algorithm needs that information to feed into Kalman Filter.

We conducted experiments using the above approach and it shows that our modified BART algorithm can achieve 87.36% accuracy in estimating network bandwidth.

2.4.3 Asymmetric Rendering Optimization Algorithm

From the above sections, it can be seen that by applying the impairment functions (Equations (2.1)$\sim$(2.5)), delay prediction model (Equations (2.12)$\sim$(15)) and relationship

between graphics rendering factors and video bitrate (Equations (2.10)∼(2.11)), we can expand the optimization function and delay constraint from the problem formulation (Equation (2.8)) to depend only on the four rendering factors P_1, P_2, TD_1, TD_2 and the current network bandwidth BW. Hence, for any time interval T_{INT}, given the measured network bandwidth BW, we can solve the optimization problem (8) to produce the optimal values for the four rendering factors, which when selected to render the left and right views will maximize user experience.

Because the rendering factors are discrete variables and our objective function has root terms, our problem can be categorized as having a discrete variable, non-linear objective function, with unequal non-linear constraint function. Since the solution space consists of four variables, with two (TD_1 and TD_2) having only 3 possible values, a branch and bound method can be efficient and feasible [62]. We describe next our branch and bound based algorithm Asymmetric Rendering Adaptation (ARA) shown in Algorithm 1.

We first define term *subproblem* as minimizing a target function (in our case minimizing I) given a set of variable ranges (in our case P_1, P_2, TD_1 and TD_2). Our aim is to solve the *original_problem* (as defined in Equation (2.8)). The *original_problem* or a *subproblem* can be further divided into *subproblems*, which has the same objective function but with a smaller variable range. The algorithm maintains a *subproblem_queue* to manage all of the *subproblems* and use ${I^{OPT}}_{temp}$, ${P_1^{OPT}}_{temp}$, ${P_2^{OPT}}_{temp}$, ${TD_1^{OPT}}_{temp}$, ${TD_2^{OPT}}_{temp}$ to store temporary optimized values. In detail, the algorithm will first initialize the upper and lower bounds for the variable ranges, set the number of iterations to be 0, set $S_P(t)$ to be 0 and set the temporary optimized I, ${I^{OPT}}_{temp}$, to be infinite. It will also put the *original_problem* into the *subproblem_queue*. Next, in step 2, the while loop continues if the length of the *subproblem_queue* is larger than 0 and the number of iteration is less than a maximum iteration threshold which is predefined. Inside the

Algorithm 1: Asymmetric Rendering Adaptation (ARA)

Input : Network bandwidth BW
Output : I^{OPT}, P_1^{OPT}, P_2^{OPT}, TD_1^{OPT}, TD_2^{OPT}
Step 1: Initialize upper bound and lower bound of P_1, P_2, TD_1, TD_2; set iteration to be 0; set $S_P(t)$ to be 0; set ${I^{OPT}}_{temp}$ to be infinite;

$$inqueue(original\_problem, subproblem\_queue)$$

Step 2:
while $(length(subproblem\_queue) > 0$ && $iteration < MAX\_INTERATION)$ **do**

$$iteration = iteration + 1$$

$$subproblem = outqueue(subproblem\_queue)$$

Step 3: $[P_{1temp}, P_{2temp}, TD_{1temp}, TD_{2temp}, I_{temp}] = s =$ $non\_linear\_continuous\_variable\_optimization(I)$
update $S_P(t)$
Step 4:
if $s =$ *Integer solution* **then**
  if $I_{temp} < {I^{OPT}}_{temp}$ **then**
    ${I^{OPT}}_{temp} = I_{temp}$
    ${P_1^{OPT}}_{temp} = P_{1temp}$, ${P_2^{OPT}}_{temp} = P_{2temp}$
    ${TD_1^{OPT}}_{temp} = TD_{1temp}$, ${TD_2^{OPT}}_{temp} = TD_{2temp}$
  end
else
  if $s =$ *Non-integer solution* **then**
    for $i = 1; i < 2^{number\_of\_non\_integer\_values}; i{+}{+}$ **do**
      Determine the ith variable range for $subproblem_i$
      $inqueue(subprobelm_i, subproblem\_queue)$
    end
  end
end
if $s =$ *No feasible solution* **then**
  continue
end
end
Step 5: Return $I^{OPT} = {I^{OPT}}_{temp}$, $P_1^{OPT} = {P_1^{OPT}}_{temp}$, $P_2^{OPT} = {P_2^{OPT}}_{temp}$, $TD_1^{OPT} = {TD_1^{OPT}}_{temp}$, $TD_2^{OPT} = {TD_2^{OPT}}_{temp}$

while loop, during every iteration, we get a *subproblem* from the *subproblem_queue*. In step 3, we relax the discrete variable optimization problem into continuous variable optimization problem and solve it using a non-linear continuous variable optimization algorithm (in our case, we use sequential quadratic programming), and denote the solution as s. After that, we update $S_P(t)$. In step 4, we have three conditions according to the output from step 3. If there is no feasible solution, we skip this branch. If it is an integer variable solution, we compare the corresponding impairment result, I_{temp}, with the current temporary optimized I, ${I^{OPT}}_{temp}$ and see if we need to replace the current temporary optimized values with s. If it is a continuous variable solution, we will further branch this *subproblem* for each non-integer value into two *subproblems* and then put them into *subproblem_queue*. For example, if $s = [10, 20, 1.5, 2, 50]$, we define $subproblem1$ to be the same as the current *subproblem* except that it will add one more constraint that $TD_1 \leq 1$. $subproblem2$ will also be the same as *subproblem* but with the constraint that $TD_1 \geq 2$. Thus, if we have m non-integer values, we will have to solve 2^m *subproblems*. Finally when the while loop terminates, we get the optimized values from temporary optimized value and finish the algorithm.

We have implemented the algorithm ARA in C; the average running time is 153ms on an Intel Xeon E5-2670 @2.60GHz processor with 15GB memory. Hence ARA can be applied in real time in every time interval T_{INT} (1 second in our current implementation). For a CMG(3D) gaming session, our overall approach is the following. In each time interval T_{INT}, we use the modified BART algorithm to measure the bandwidth BW (Section 2.4.3), and use BW as input to ARA to get the optimal asymmetric rendering factors for the time interval. The above is repeated for each time interval, leading to dynamic adaptation of left and right view rendering factors optimal to the changing network bandwidth, so as to maximize overall user experience during the entire CMG(3D) session.

2.5 Experimental Results

In this section, we report on experiments conducted using a commercial cloud service to verify the performance improvement possible by applying the proposed Asymmetric Rendering Adaptation technique. We use the same testbed as shown in Figure 2.3, except we implement our CMG(3D) system, including the ARA algorithm, on Amazon Web Service (AWS) cloud servers. Specifically, we use AWS g2.2xlarge instance which provides access to one NVIDIA GRID GPU with 1,536 CUDA cores and 4GB of video memory. The CPU it provides is Intel Xeon E5-2670 @2.60GHz with 15GB memory. The operating system we deploy is Windows_Server-2008-R2_SP1. As explained before, we emulate network traces collected from a 4G-LTE network using the network emulator in the testbed.

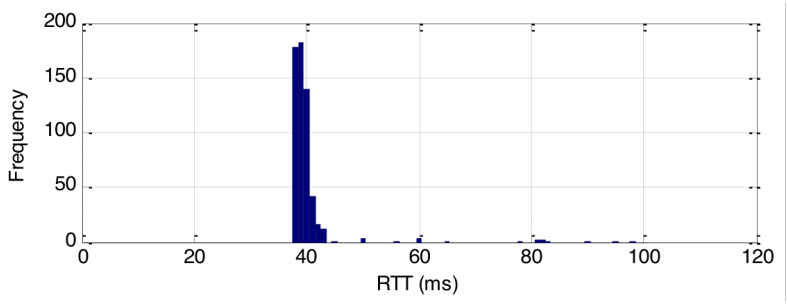


Figure 2.14. RTT measured from AWS to test device

Firstly, in order to determine the propagation delay, $D_P(t)$, in Equation (2.12) from AWS cloud server to the 3D device, we performed an experiment to record the round trip delay (RTT) at different times of a day. We use Ping to test for RTT, with testing every 10 minutes from 8:00 a.m. to 10:00 p.m., Monday to Sunday, to get 588 data points in total. Figure 2.14 shows the PDF of the results. We find that the RTT between AWS cloud server to the 3D device in our testbed is very stable-mostly between 38ms to 41ms with very limited number of exceptions which have longer RTTs. According to the above, we calculate the average RTT to be 40.2ms. Thus, considering the downlink

delay to be half of RTT, we use 20.1ms value for $D_P(t)$ in Equation (2.12). In addition, in order to determine $D_S(t)$, we measured the rendering and encoding time on AWS and found out it was on average 2ms for rendering and encoding one frame. Thus we set $D_S(t)$ to be 2ms in the following experiments.

We compared three approaches using the testbed: 1) Asymmetric Rendering Adaptation (ARA) which adapts the rendering of left and right views depending on the network conditions using our proposed approach, 2) Symmetric Rendering Adaptation (SRA) which is an adaptation algorithm for cloud mobile 2D display gaming introduced in [51] where the rendering factors of left and right views are adapted symmetrically, and 3) No adaptation (NA) in which we fix texture detail to be medium and view distance to be 150m for both views. For fair comparison between ARA and SRA, we disable video encoding adaptation option for SRA, but rather fix it to produce high video quality using $QP = 25$ like in the case of ARA.

In addition, in order to verify our model and optimization algorithm can be applied to different kinds of games, we tested two different games of different genres based on the above testbed. Figure 2.15 shows the results for Planeshift and Figure 2.16 shows the results for Broadsides [63]. For both games, we show step by step results in the following order.

Figures 2.15a and 2.16a show the 4G-LTE mobile network bandwidth profile we captured and then used (emulated using the network emulator) when playing the games for all the three approaches. Figures 2.15b, 2.15c, 2.15d 2.16b, 2.16c and 2.16d show the graphics rendering factors used when playing the games. We can see that for both games, while NA cannot adapt to the network bandwidth, ARA and SRA can both choose high texture quality and large view distance when network condition is good but low texture quality and small view distance when network bandwidth is tight. However, because ARA can use more choices (separate TD and P for each view), we can see

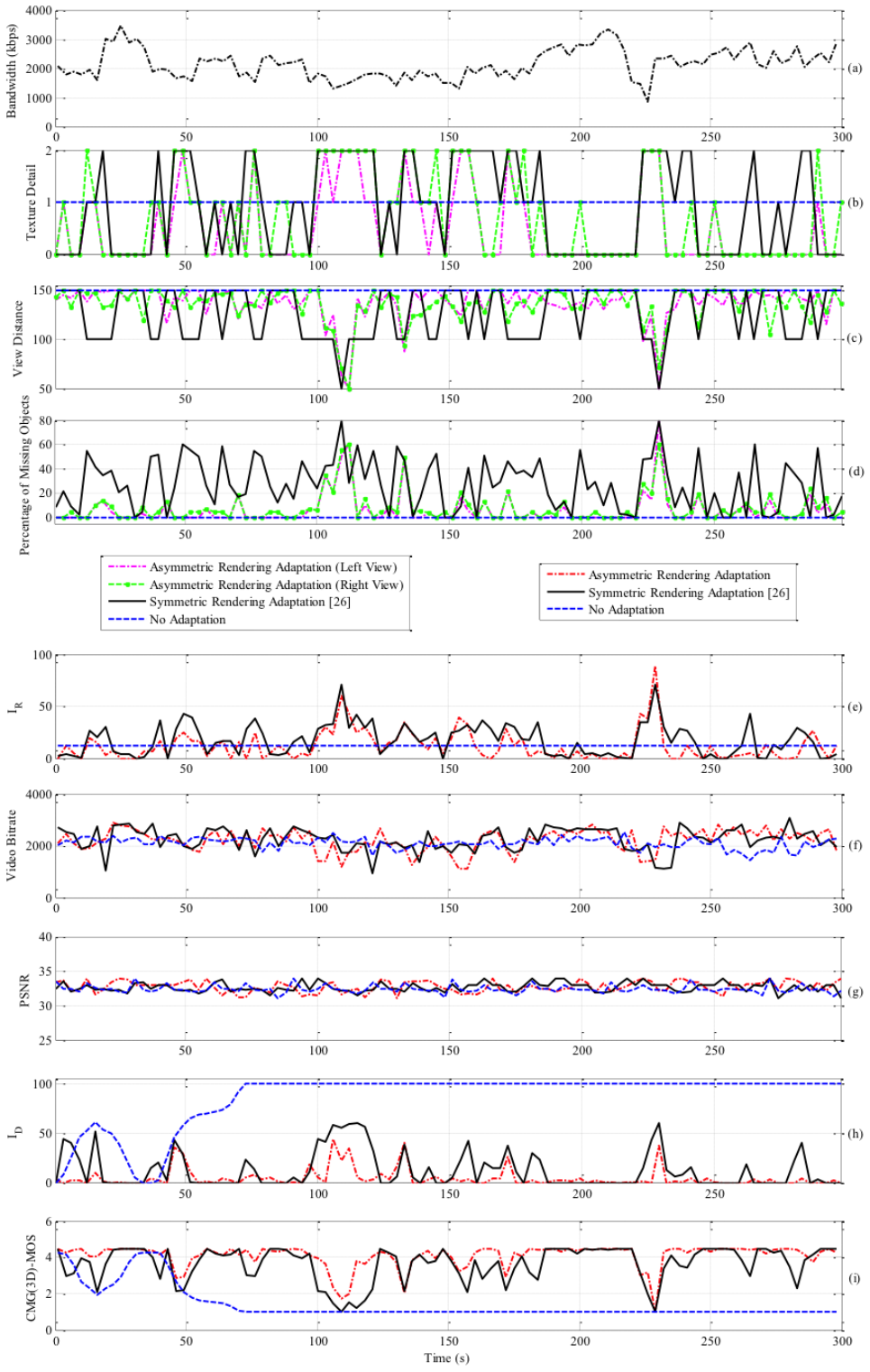


Figure 2.15. Results for game PlaneShift

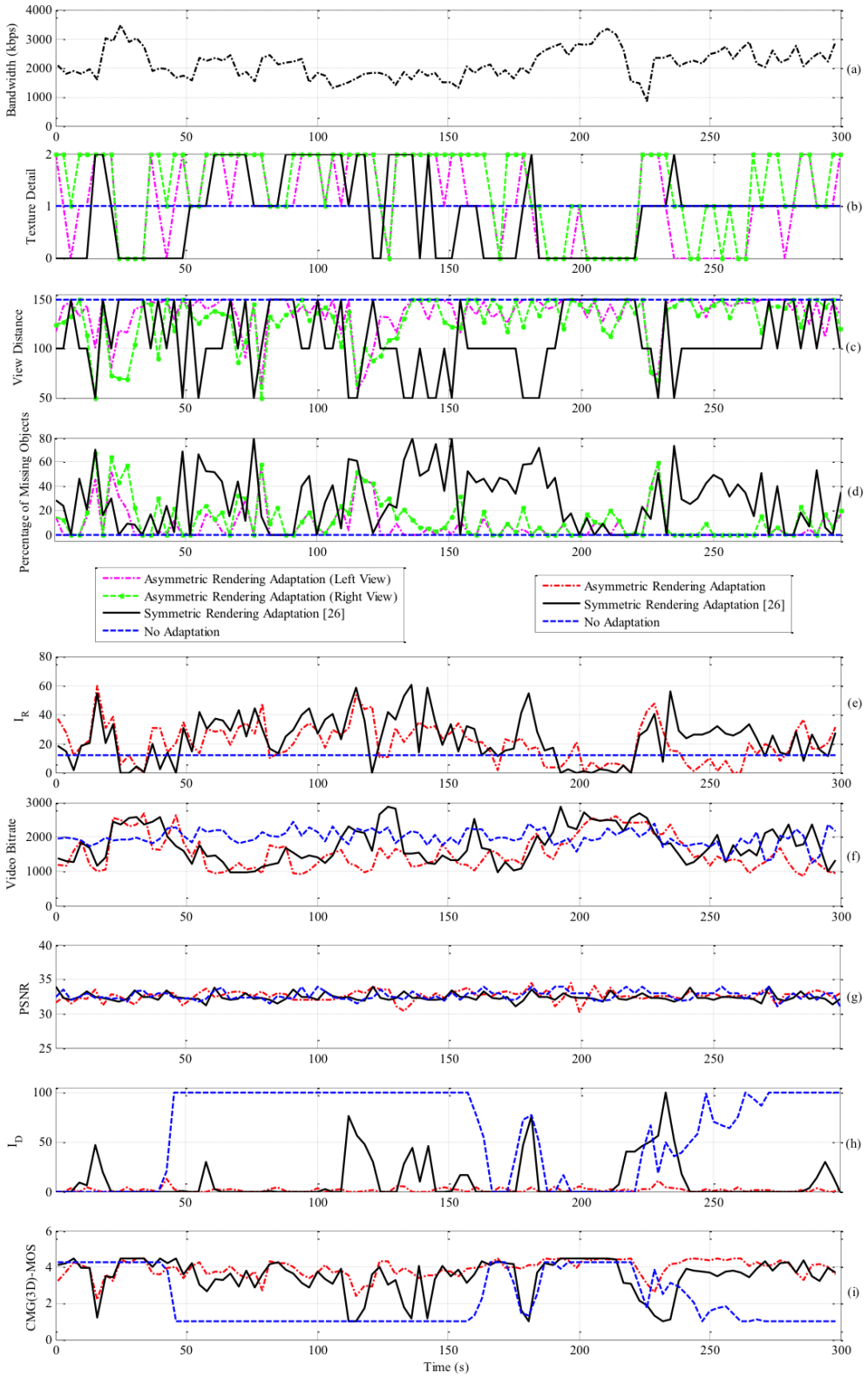


Figure 2.16. Results for game Broadsides

that ARA can choose better combinations (higher values) of texture detail and view distance than SRA. Figures 2.15e and 2.16e show the value of rendering impairment, IR, calculated by our model for all three approaches. For the game Planeshift, the average I_R is 12.57 for ARA, 18.32 for SRA and 12.19 for NA, and for the game Broadsides, the average I_R is 20.80 for ARA and 23.48 for SRA and 12.19 for NA. Figures 2.15f and 2.16f show the actual bitrate of the video encoded. In both plots, the video bitrate show great correlation with the graphics rendering factors which means that when the graphics rendering factors are using high texture quality and large view distance, the bitrate rises up and when low texture detail and small view distance are applied, the bitrate is kept at a low level. Figure 2.15g and 2.16g show the PSNR for the resulting encoded video. Because we are using constant QP to encode, we see that a high and stable value of PSNR is maintained as desired. For the game Planeshift, the mean PSNR for ARA, SRA, and NA are 32.65, 32.65 and 32.39 respectively. For the game Broadsides, they are 32.70, 32.24 and 32.4. Figure 2.15h and 2.16h show ID which is calculated using actual delay measured. We can see that ARA can maintain low delay much better than the other two strategies. Figures 2.15i and 2.16i show CMG(3D)-MOS which takes both I_R and I_D into consideration. The mean CMG(3D)-MOS for the game PlaneShift are 4.09, 3.52 and 1.41 using ARA, SRA and NA respectively. Similarly, for the game Broadsides, the mean CMG(3D)-MOS for ARA, SRA and NA are 3.94, 3.46 and 2.19 respectively.

To further prove the advantage of ARA, we performed a final round of subjective tests which includes 17 UCSD students (12 males, 5 females; aged 19$\sim$27). We used the same network profile as is shown in Figure 2.15a. We divided the whole 5 minutes testing into 5 time segments, 1 minute each. The testers were asked to give a score according to Table 2.5 at the end of each 1-min segment for each adaptation algorithm. Table 2.7 and Table 2.8 list the minute-wise average subjective CMG(3D)-MOS for the three algorithms with 95% confidence interval range. (For example, “3.560.04” means

the average is 3.56 and the 95% confidence interval is 3.52-3.60.) For game PlaneShift, the average CMG(3D)-MOS for the three adaptation algorithms are 3.48, 3.15 and 1.42, using ARA, SRA, and NA respectively. For game Broadsides, the average CMG(3D)-MOS for the three adaptation algorithms are 3.69, 3.35 and 1.96, using ARA, SRA, and NA respectively. Note that the average 95% confidence interval range for two games is only 0.082 which indicates that the results given by the subjects are statistically valid. From the tables, we can observe that for game Planeshift, ARA outperforms SRA by up to 26.2% (segment 3) and on average 10.5%. For game Broadsides, ARA outperforms SRA by up to 33.8% (segment 3) and on average 10.1%.

Table 2.7. Subjective test results to compare different adaptation algorithms using game PlaneShift

Time Segment	1	2	3	4	5
ARA	3.56±0.04	3.78±0.03	3.08±0.06	2.98±0.05	4.02±0.03
SRA	3.42±0.04	3.45±0.06	2.44±0.07	2.82±0.08	3.64±0.05
NA	3.12±0.05	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0

Table 2.8. Subjective test results to compare different adaptation algorithms using game broadsides

Time Segment	1	2	3	4	5
ARA	3.68±0.05	4.08±0.04	3.28±0.03	3.54±0.03	3.89±0.04
SRA	3.55±0.06	3.89±0.08	2.45±0.1	3.18±0.05	3.67±0.06
NA	3.32±0.05	1.0±0.0	1.0±0.0	3.46±0.09	1.0±0.0

The above results demonstrate the significant advantage of our proposed asymmetric graphic rendering adaptation approach (ARA) to deliver consistent and high user experience on 3D displays when playing Cloud based 3D games, in spite of dynamically changing and challenging mobile network conditions.

2.6 Conclusions

In this chapter, we propose an asymmetric graphics rendering technique for Cloud Mobile 3D Display Gaming, in order to address the challenge of delivering 3D rendered video with high user experience over fluctuating and constrained wireless networks. To study the feasibility of this asymmetric graphics rendering technique, we developed and validated a user experience model to quantitatively measure user experience, including impairments due to asymmetric graphics rendering and network delay, using extensive subjective experiments. Moreover, based on the CMG(3D)-UE model, we proposed a technique to automatically select the optimal graphics rendering factors for each of the views, according to the changing network conditions. Our experiments conducted using real cellular network and Amazon Cloud Servers demonstrate that our proposed technique can achieve much better CMG(3D)-MOS than techniques proposed before [51] and is applicable and effective to 3D games of different genres.

In the next chapter, we will extend the idea of asymmetric graphics rendering to let each individual object to choose its own texture detail so that objects of higher importance will be set of higher quality and unimportant objects can sacrifice more. In this way, the overall user experience can be improved given limited bit rate budget.

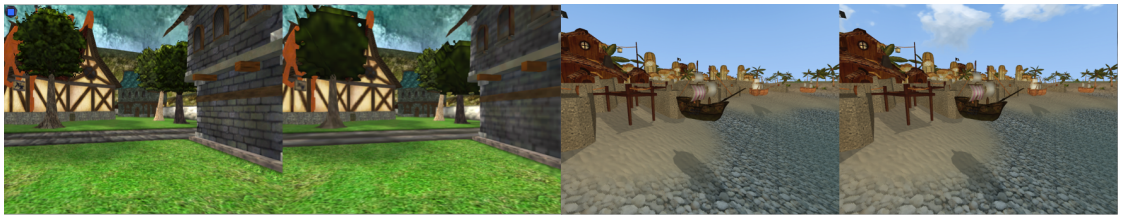
2.7 Acknowledgements

Chapter 2, in part, is from the material as it appears in proceedings of IEEE ICNC 2014. Yao Lu; Yao Liu; Sujit Dey. and in IEEE Journal of Selected Topics in Signal Processing 2015. Yao Lu; Yao Liu; Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Asymmetric and Selective Object Rendering

3.1 Introduction



(a) **(b)**

Figure 3.1. (a) left, Example of asymmetric object rendering, (b) right, Example of selective object rendering

In the last chapter, we proposed the idea of asymmetric graphics rendering. However, all objects need to be of the same texture quality in one view. Inspired by region-of-interest (ROI) based video encoding [68, 69] which allocates more encoding bits to the more important regions of a video frame and fewer bits to the less important ones and therefore reduce the bit rate for the same perceptual quality, we propose a similar approach for graphics rendering. While rendering a frame, we can increase the richness of rendering for the more important objects while decreasing the rendering richness for other less important objects, thereby potentially reducing the video content

complexity and hence the video bit rate needed to encode the rendered frame while maintaining the perceived video quality. We call this technique Object Rendering. The term richness of rendering includes several factors such as texture detail, realistic effect, scene complexity, etc. In this chapter we focus on texture detail, and the other factors could be researched later as future work. The texture is defined as an image used to put on top of the game objects when doing rendering. We define texture detail to be High(H) when the game is using the original texture images, to be Medium(M) when the texture images are down-sampled once, and Low(L) when the texture images are down-sampled twice. In addition, because our use case is true 3D gaming where we will render two views, and these two views will be seen by the user's two eyes separately, we propose a technique called Asymmetric Object Rendering, where the rendering richness of an object for one view can be different from the rendering richness of the same object for the other view. Figure 3.1a shows an example, where the left view (left image) has all the objects rendered with Medium texture detail, while in the right view, the tree on the left side and right side, the building on the right edge, and the path are rendered with Low texture detail but all the other objects are rendered with Medium texture detail.

In addition, also inspired by the preliminary work done by Hemmati et al. [50], we found that for each object, not only could we set specific texture detail for it, but we could also choose to not render it at all. Figure 3.1b shows an example. The two images are both for the left view of the game. However, the left image is not rendered with clouds while the right image is rendered with clouds. We combine these two techniques in this chapter and call it Asymmetric and Selective Object Rendering (ASOR).

In the last chapter, we have shown that by lowering texture detail, or not rendering some objects, video bitrate can be lowered tremendously. However, ASOR influence user experience, thus we need to develop a model to quantitatively measure the user experience when the proposed ASOR is performed.

3.2 User Experience Model

We define Cloud Mobile 3D Display Gaming Mean Opinion Score (CMG(3D)-MOS) as a measurement metric for CMG(3D)-UE. To derive the formula for CMG(3D)-MOS with ASOR, we follow the same framework initially proposed by ITU [58] and modified by us in our previous work, where the user experience is formulated using a set of impairment functions including the impairment functions for graphics rendering (I_R), video encoding (I_V) and network (I_N).

$$\text{CMG(3D)-MOS} = 1 + 0.035R + 7 \times 10^{-6} R\,(R-60)\,(100-R) \tag{3.1}$$

$$R = 100 - I = 100 - f\,(I_R, I_V, I_N) \tag{3.2}$$

In Equation (3.1), the CMG(3D)-MOS metric is formulated by a transmission rating factor R, which represents the overall user experience. R factor takes value between 0 and 100; higher R value corresponds to higher CMG(3D)-MOS and better user experience. In Equation (3.2), the term I stands for the combined impairment (described by function f) caused by various factors including graphics rendering (I_R), video encoding (I_V) and network factors(I_N). However, in this work, we will only need to consider impairment caused by rendering. We will set sufficiently high encoding quality on the encoder (for example, set a low quantization parameter, QP), so there will be no impairment due to encoding. Also, we assume there will be no impairment due to the wireless network, as we can adapt the rendering setting effectively such that the resulting video bit rate will not exceed the available bandwidth and cause network congestion. Thus, the formula of R factor (Equation (3.2)) can be re-stated as:

$$R = 100 - I_R \tag{3.3}$$

in which I_R is the graphics rendering impairment.

Furthermore, in the proposed ASOR approach, we will both decide whether or not to render the object and adapt the rendering richness (more specifically, texture detail) for each individual object depending on its importance; hence, we propose the following equation to model I_R

$$I_R = f(I_S, I_{NS}) \tag{3.4}$$

$$I_S = \sum_{i=1}^{K} w_i p_i I_{TD}(TD_{L\_i}, TD_{R\_i}) \tag{3.5}$$

$$I_{NS} = g\left(\sum_{i=1}^{L} w_i p_i\right) \tag{3.6}$$

with the constraints of

$$\sum_{i=1}^{K} w_i p_i = 1 \tag{3.7}$$

$$\frac{w_1}{w_2} = c_1, \ldots, \frac{w_i}{w_{i+1}} = c_i, \ldots, \frac{w_{K-1}}{w_K} = c_{K-1} \tag{3.8}$$

In Equation (3.4), I_S means the impairment caused by changing texture detail of objects which are selected to be rendered while I_{NS} stands for the impairment caused by (not rendering) the objects which are not selected to be rendered. The total impairment I_R is a combination of these two impairments (I_S and I_{NS}) described by a function f which we will derive later through subjective tests. In Equation (3.5), K represents the total number of objects that are rendered in the game frame and I_{TD} function describes different impairment values caused by the different texture detail settings on the left and right views. The parameter i is the object index, p_i means the percentage of pixels that the object i occupies within a game frame and w_i means the object weight which is related to the importance of the object i. Equation (3.5) can be interpreted as: the overall impairment caused by the objects that are selected to be rendered, I_S is a weighted average of impairment caused by each object ($I_{TD}(TDL_i, TDR_i)$), considering the importance of

this object (w_i) and how much space this object occupies in the game frame (p_i). The value of p_i can be conveniently extracted from the game engine and the importance of each object w_i is a fixed parameter for each object and can be derived through subjective test. In Equation (3.6), L represents the total number of objects that are not selected to be rendered. We define the impairment I_{NS} to be a function of the sum of the product of the importance of each object i, w_i, and how much space this object i occupies in the game frame. The function g will be derived through subjective tests. Equation (3.7) is a constraint function which normalizes the weight for object i, w_i. Equation (3.8) is defined so that the importance relationship (ratio) between any two object in one scene is fixed (defined by fixed values $c_1, \ldots, c_{K-1}$). The rationale behind these two constraint functions is that we want to make sure I_S computed by Equation (3.5) will not be too large or too small at the same time that the importance between any two objects should be maintained and contribute to the I_S.

To summarize, we propose Equations (3.4) to (3.8) as a framework to model the impairment caused by graphics rendering as I_R. In this framework, first, all the objects will be divided into two groups. One group of the objects is decided to be rendered whose impairment value I_S can be computed by Equation (3.5). The other group of objects is decided not to be rendered and exclude from the game frame whose impairment value I_{NS} will be computed by Equation (3.6). For those objects that are rendered, TD_{L_i} and TD_{R_i} for each object i will be set and the pixel that each object i occupies, p_i can be extracted from game engine. Because pi is different in each game frame, by using Equations (3.7) and (3.8), w_i associates with p_i can be computed. Thus, TD_{L_i}, TD_{R_i}, w_i and p_i can be regarded as input values and our research goal is to figure out how to derive I_{TD} function, the values of $c_1, \ldots, c_{K-1}$, the function g and the function f to complete the model and output the final impairment I_R according to the input values.

In the following, we will first propose the procedures and theories of how to

derive I_{TD} function, the values of $c_1, \ldots, c_{K-1}$, the function g and the function f. Then, we perform subjective tests to get data to derive the parameters in the models and finally we validate the model through another set of subjective test results.

3.2.1 Impairment Function Derivation

Firstly, to derive I_{TD} function, we design our experiment (Exp_A) as follows.

We render all the objects and change texture details for all the objects together, making all of them to be the same. In this case, because I_{TD} value for each object is the same and because of the constraint Equation (3.7), Equations (3.4) to (3.6) can be re-stated as

$$I_R = I_S = (\sum_{i=1}^{K} w_i p_i) I_{TD}(TD_L, TD_R) = I_{TD}(TD_L, TD_R) \tag{3.9}$$

By collecting the subjective test scores from users, we can easily get the formulation of I_{TD} function. Because the combination of the texture detail for both views is limited (5 in total), we consider to use a lookup table to represent I_{TD} function.

Secondly, to derive the values of $c_1, \ldots, c_{K-1}$, we design the experiment (Exp_B) in another way as follows.

We still render all the objects. We vary the texture detail settings for each object one at a time (5 combinations for each object and we have K objects so $5K$ times in total) and then collect p_i of each object and I_S (impairment scores given by subjects) associated with each round.

For each round, we can derive the following:

1) By plugging Equation (3.8) into Equation (3.7), we can get:

$$\begin{aligned} & w_1 p_1 + w_2 p_2 \ldots + w_K p_K = 1 \\ \Leftrightarrow\ & w_1 p_1 + \frac{w_1}{c_1} p_2 + \ldots + \frac{w_1}{c_1 c_2 \ldots c_{K-1}} p_K = 1 \end{aligned} \tag{3.10}$$

2) By plugging Equation (3.8) into Equation (3.6), we can get:

$$\begin{aligned} & w_1 p_1 I_{TD}(TD_{L\_1}, TD_{R\_1}) + w_2 p_2 I_{TD}(TD_{L\_2}, TD_{R\_2}) + \ldots \\ & + w_K p_K I_{TD}(TD_{L\_K}, TD_{R\_K}) = I_S \\ \Leftrightarrow\ & w_1 p_1 I_{TD}(TD_{L\_1}, TD_{R\_1}) + \frac{w_1}{c_1} p_2 I_{TD}(TD_{L\_2}, TD_{R\_2}) + \ldots \\ & + \frac{w_1}{c_1 c_2 \ldots c_{K-1}} p_K I_{TD}(TD_{L\_K}, TD_{R\_K}) = I_S \end{aligned} \tag{3.11}$$

3) Let $I_{TD}(TD_{L\_i}, TD_{R\_i}) = I_i$, let $1/(c_1 c_2 \ldots c_i) = z_i$. Combining Equation (3.10) with Equation (3.11) we can get:

$$w_1 p_1 I_S + \frac{w_1}{c_1} p_2 I_S + \ldots + \frac{w_1}{c_1 c_2 \ldots c_{K-1}} p_K I_S = w_1 p_1 I_1 + \frac{w_1}{c_1} p_2 I_2 + \ldots + \frac{w_1}{c_1 c_2 \ldots c_{K-1}} p_K I_K \tag{3.12}$$

4) Let $1/(c_1 c_2 \ldots c_i) = z_i$, we can get:

$$\begin{aligned} & p_1 I_S + z_1 p_2 I_S + \ldots + z_{K-1} p_K I_S = p_1 I_1 + z_1 p_2 I_2 + \ldots + z_{K-1} p_K I_K \\ \Leftrightarrow\ & p_1 I_S - p_1 I_1 + z_1 (p_2 I_S - p_2 I_2) + \ldots + z_{K-1} (p_K I_S - p_K I_K) = 0 \end{aligned} \tag{3.13}$$

Then, by doing $5K$ rounds of experiments, we can get $5K$ implementations of Equation (3.13). Next, Let $p_i I_S - p_i I_i$ for round m be L_{i_m} we can get the following

matrix:

$$\begin{pmatrix} L_{1\_1} & \cdots & L_{K\_1} \\ L_{1\_2} & \cdots & L_{K\_2} \\ \vdots & \ddots & \vdots \\ L_{1\_5K-1} & \cdots & L_{K\_5K-1} \\ L_{1\_5K} & \cdots & L_{K\_5K} \end{pmatrix} \begin{pmatrix} 1 \\ z_1 \\ z_2 \\ \vdots \\ z_{K-1} \end{pmatrix} = 0 \tag{3.14}$$

Notice that Equation (3.14) is an over-determined linear equation. Thus, a least square method can be applied to solve $z_1, \ldots, z_{K-1}$. Note that theoretically, K rounds of experiments will provide enough data to solve the equation. However, because the subjective tests scores have some variations and are therefore noisy, it is recommended to do more rounds of experiments to fit the data and get higher accuracy.

After getting the values of $z_1, \ldots, z_{K-1}$, they can be easily translated back into the values of $c_1, \ldots, c_{K-1}$.

Thirdly, The derivation of g function and f function can both be considered as one dimensional data regression problem. We will use regression techniques to derive these functions.

3.2.2 Subjective Test settings

Figure 3.2. Testbed for subjective experiments

Figure 3.2 shows the testbed used for the subjective tests. We use a 3D monitor with a laptop to substitute for 3D display of mobile devices because current available mobile 3D displays do not have as good quality as 3D monitors that may cause additional impairment which we want to avoid. The laptop is connected to a network emulator via

an Access Point and the network emulator is connected to the game server. We select two games of different genres, enable ASOR by reprogramming the rendering loop in the rendering engine and perform the subjective tests. The first game is a first person shooting game Broadsides [63] and the second game is a Massively Multiplayer Online Role-Playing Game Planeshift [57]. To investigate how ASOR affect user experience, we set the video encoding parameter QP to 25 which leads to sufficiently high encoding video quality and set network bandwidth to be sufficiently large so that only texture detail can cause impairment. We then invited 23 UCSD students (13 male, 10 female; aged 18-28) to participate in our subjective experiments. Firstly, we asked the testers to sit before a 23 inch LG D2342 3D Monitor, and show them a 3D video as a training sequence before the real test starts to let the testers adjust their viewing angle. After that, we start the game and set the graphics rendering factors manually according to the rules that will be explained in the next subsection. Once a combination of rendering factors is set, we ask the testers to play the game for 1 minute and evaluate the graphics rendering impairment according to the criterion listed in Table 3.1. During the whole experiment, the testers were asked to control the avatar to perform multiple tasks (including attacking an enemy boat, sailing towards an island, etc. for Broadsides and looking for an object, talking to a Non-player character (NPC), attacking an enemy etc. for Planeshift).

Table 3.1. Scoring criterion for rendering impairment I_R

I_R	**Description**
0	No visual impairment
0-20	Minor visual impairment
20-40	Noticeable visual impairment
40-60	Clear visual impairment
60-100	Unacceptable visual impairment

Table 3.2. Subjective test results: average I_{TD} scores for different texture detail combinations in different scenes

Texture Detail Combination	H-H	H-M	H-L	M-M	M-L	L-L
$I_{TD\ 1\ Broadsides}$	0	18.52	43.35	22.5	24.45	40.5
$I_{TD\ 2\ Broadsides}$	0	17.35	40.25	23.24	26.23	38.74
$I_{TD\ 3\ Broadsides}$	0	18.34	45.32	25.17	27.12	39.27
$I_{TD\ 1\ Planeshift}$	0	7.30	27.85	10.38	19.84	25.38
$I_{TD\ 1\ Planeshift}$	0	8.30	28.85	11.61	20.46	24.00
$I_{TD\ 1\ Planeshift}$	0	7.84	25.61	12.30	20.76	23.07

3.2.3 Model Parameter Derivation and Validation

There are multiple genres of games and within one game there are different game scenes with different spatial and temporal characteristics; hence the entire rendering impairment caused by texture detail and selective rendering may be affected by the type of the game and even by different game scenes in the same game. To gain better understanding and ensure that the impairment functions derived are general and applicable to most of games and are consistent across different game scenes, in the following, we select three different scenes in each Broadsides and Planeshift to conduct our subjective study. In addition, for simplicity but without the loss of generality, in our experiments, we will discuss a special scenario, where all the objects can be divided into two groups and each group of objects has the same importance. We define the following two categories of objects: 1) Key Objects (KO) and 2) General Objects (GO). KO is defined as the objects which are important to the players and will attract a lot of attention. For example, in the game Broadsides, the player's boat and enemy boats are KO. GO is defined to be the objects which do not influence the execution of game logic, and eliminating them will not severely affect the playability of the game. For example, trees, rocks, ports, etc. Thus, by replacing w_i with w_{KO} and w_{GO}, Equation (3.5),(3.7),(3.8) can be rewritten as

the following.

$$I_S = \sum_{i=1}^{K_1} w_{KO} p_i I_{TD}\left(TD_{L\_i}, TD_{R\_i}\right) + \sum_{j=1}^{K_2} w_{GO} p_j I_{TD}\left(TD_{L\_j}, TD_{R\_j}\right) \tag{3.15}$$

with the constraints of

$$\sum_{i=1}^{K_1} w_{KO} p_i + \sum_{j=1}^{K_2} w_{GO} p_j = 1 \tag{3.16}$$

$$\frac{w_{KO}}{w_{GO}} = c \tag{3.17}$$

In Equation (3.15), K_1 and K_2 represent the total number of KO and GO objects respectively, w_{KO} and w_{GO} mean the importance weight for KO and GO respectively.

Based on the above setup and assumptions, we will derive the model parameters in the following.

Firstly, we derive I_{TD} function. According to Equation (3.9) and by performing EXP_A, the scores given by the subjects to evaluate the overall rendering impairment is equal to the value computed by $I_{TD}(TD_L, TD_R)$ function. We collected the results for Exp_A and compute the average scores. The results are listed in Table 3.2. Note that the results of Planeshift are referenced from our previous work.

From Table 3.2 we can make the following observation. For each game and each combination of texture detail, the average impairment score given by the testers are very close to each other irrespective of the scene, with an average standard deviation of 0.97; hence we believe that for a specific game, the complexity of the scene does not influence the score too much, instead, the texture detail dominates the impairment. However, we also find that the scores are very different between two games, which indicates that the impairment caused by texture detail maybe related to the genre of the game.

Secondly, because the model is simplified, instead of computing model parameters $c_1, \ldots, c_{K-1}$ in Equation (3.8), we only need to compute c in Equation (3.17). Thus,

Equation (3.14) can also be re-written as:

$$c = \frac{\sum_{j=1}^{K_2} p_j I_{TD}(TD_{L\_GO}, TD_{R\_GO}) - I_S \sum_{j=1}^{K_2} p_j}{\left(I_S \sum_{i=1}^{K_1} p_i - \sum_{i=1}^{K_1} p_i I_{TD}(TD_{L\_KO}, TD_{R\_KO}) \right)} \tag{3.18}$$

Table 3.3. Derivation of value c for different games and in different scenes

Scene	1_B	2_B	3_B	1_P	2_P	3_P
c	1.78	2.34	1.25	4.76	5.34	2.79

Table 3.4. Derivation of model parameter for I_{NS} function

Scene	1_B	2_B	3_B	1_P	2_P	3_P
k	8.67	9.74	9.51	12.18	11.75	11.20
x_0	0.61	0.60	0.63	0.44	0.45	0.48

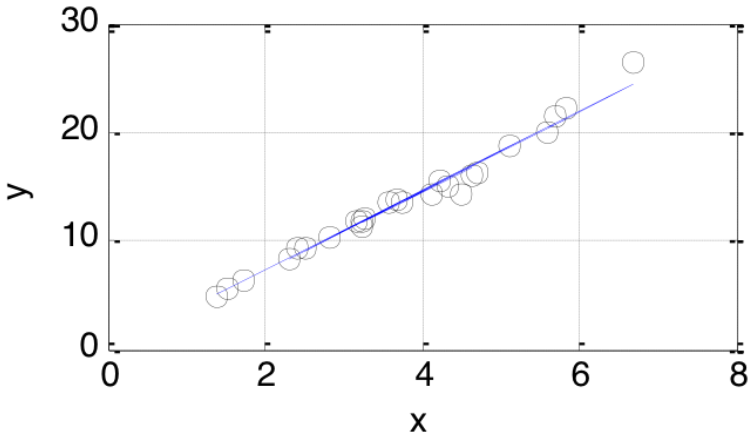


Figure 3.3. Derivation for c

In our Exp_B, we set K to be 5, so there are 25 implementations of Equation (3.18) in total and we apply least square method to solve c. Figure 3.3 shows results for scene 1 of the game Broadsides where x-axis and y-axis represent the denominator and numerator of Equation (3.18) respectively after using the scores I_S given by the subjects. For scene

1 of the game Broadsides, $c = 3.78$. The results for other scenes are listed in Table 3.3. From Table 3.3, we can see that the value c varies from game to game and even scene to scene.

Note that in the above derivation, the number of pixels p_i for each object i for each video frame is different; since a subject plays the game for 1 minute which equals to $25 * 60 = 1500$ frames, we use the average p_i value in the derivation process.

Thirdly, in order to derive the function g in Equation (3.6), we set the texture detail of all the objects that are rendered to be High. We randomly select k objects that are not to be rendered with $k = 1, 2, 3, 4, 5, 6$. For each k, we repeat the experiment for 5 times, selecting k randomly different objects each time. We consider the total set of 30 conditions to be enough to derive the function g. After obtaining the subjective scores from the subjects, we plot the results in Fig 6 with x-axis showing the sum of the product of w_i and p_i, and y-axis showing the impairment value (subjective score).

From Figures 3.4 and 3.5, we can see that the curve is very flat when the product of w_i and p_i is small, but then it goes up with a steep slope and finally becomes flat again. We aim to select a mathematical function to fit the data accurately. We try regression methods with linear function, sigmoid function, piecewise linear function and use the sum of absolute difference as the metric to select the best fitted function. We found that use of sigmoid function (Equation (3.19)) can best fit the data.

$$I_{NS} = \frac{100}{1 + e^{-k(\sum_{i=1}^{L} w_i p_i - x_0)}} \tag{3.19}$$

Figures 3.4 and 3.5 also show the fitted results and Table 3.4 shows the parameters k and x_0 derived accordingly.

From the results above we can see the parameters of different scenes for each game are very close to each other but are not close between two different games. Thus,

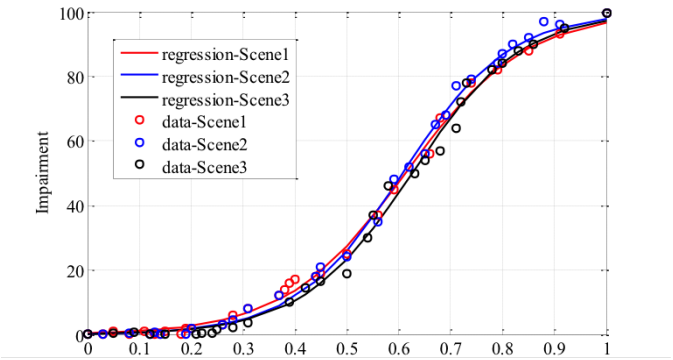


Figure 3.4. Derivation of I_{NS} using the game Broadsides

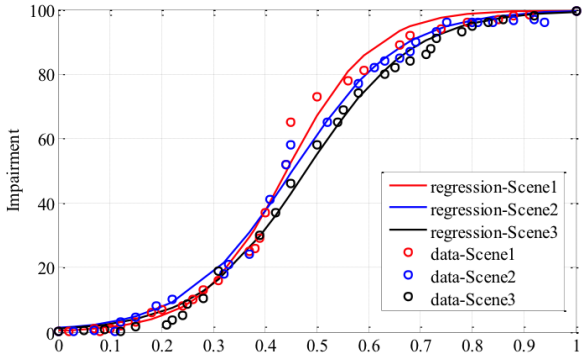


Figure 3.5. Derivation of I_{NS} using the game Planeshift

we conclude that the I_{NS} function is not scene dependent but is game dependent.

Fourthly, after getting the separate impairment functions I_S and I_{NS}, we need to derive the entire impairment function I_R. In order to achieve that, we then change the texture detail settings for each object independently, and select whether or not to render an object randomly in all six scenes and ask the testers to give scores for I_R. We use 60% of the data to derive the function f in Equation (3.4) and use the rest 40% of the date to finally validate the entire model later.

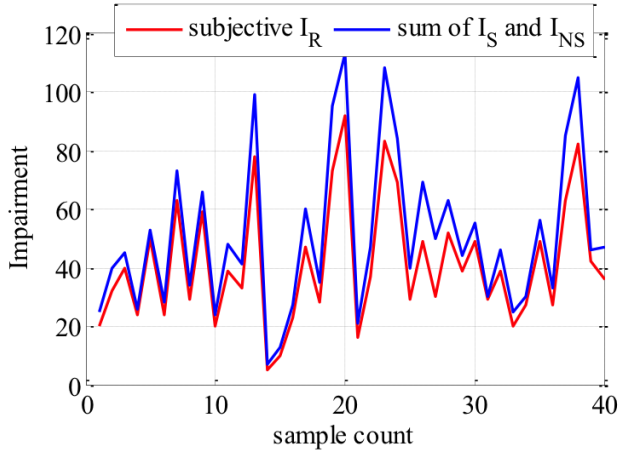


Figure 3.6. Derivation of I_R using the game Broadsides

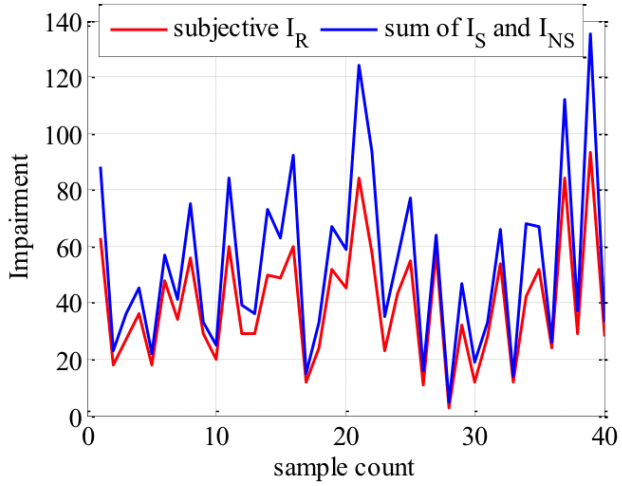


Figure 3.7. Derivation of I_R using the game Planeshift

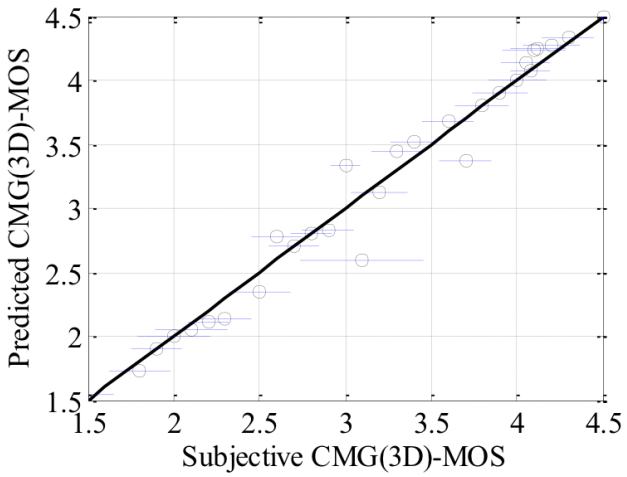


Figure 3.8. Validation of I_R with blue line showing 95% confidence interval for the game Broadsides

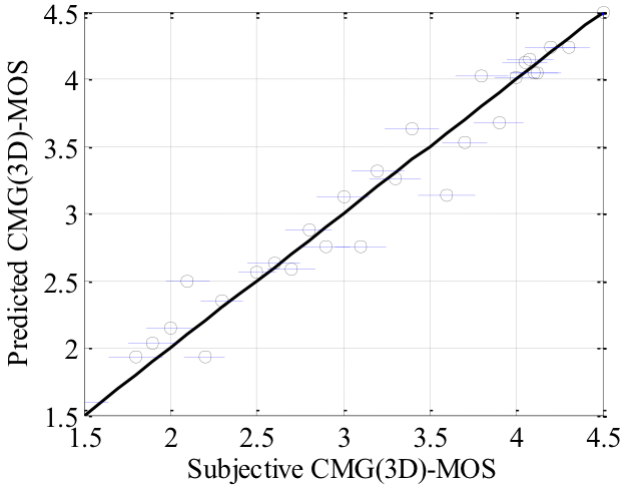


Figure 3.9. Validation of I_R with blue line showing 95% confidence interval for the game Planeshift

Figures 3.6 and 3.7 plot both I_R given by the subjects and the sum of I_S and I_{NS} calculated using the functions we derived previously for both games. We find that I_R is always bigger than the sum of I_S and I_{NS}. In some cases, it even exceeds 100 which is the maximum value we set for I_R. Thus we propose to model I_R as Equation (3.20)

$$I_R = \min\left(I_S + I_{NS} - h\left(I_S, I_{NS}\right), 100\right) \tag{3.20}$$

We tried the following possible h functions and using regression method to derive the parameter.

$$h_1\left(I_S, I_{NS}\right) = pI_S I_{NS} \tag{3.21}$$

$$h_2\left(I_S, I_{NS}\right) = p\sqrt{I_S I_{NS}} \tag{3.22}$$

$$h_3\left(I_S, I_{NS}\right) = p(I_S I_{NS})^2 \tag{3.23}$$

We use the sum of absolute difference as metric and find that using Equation (3.21) as the h function achieves the best accuracy. The parameter p derived for two games are 0.22 for Broadsides and 0.27 for Planeshift.

Finally, after deriving all the parameters and necessary functions, we use 40% of the data mentioned above to validate the derived model. Figure 3.8 is the scatter plot of the relationship between subjective and predicted CMG(3D)-UE scores for the game Broadsides and Figure 3.9 is for the game Planeshift. The correlation of Figure 3.8 is 0.986 and the correlation of Figure 3.9 is 0.974. Thus, we can conclude that the proposed IR model (including Equations (3.1),(3.3),(3.4),(3.7),(3.8),(3.19),(3.20),(3.21) and Table 3.2,3.3,3.4) will lead to high modeling accuracy.

In the above subsections, we have completed deriving a model which describes the relationship between the impairment caused by ASOR proposed in this work and the rendering settings. The high accuracy achieved by using the model for two games

of very different genre show the model can be generally applicable. However, there are several parameters that need to be derived for each game and even for each specific scene. We consider the effort needed to derive the parameters for each game to be acceptable compared to the effort it needs to develop a complete 3D game. In addition, a specific parameter, the ratio of weight between KO and GO c (Equation (3.18)), needs to be derived for each scene), which may need significant effort. However, as is the challenge for all applications based on ROI technology, how to decide the importance ratio still remains an open problem. Compared to previous works [68, 69] which assign the importance value manually or according to the features extracted from video frames such as distance, brightness, edge, etc. that is sophisticated and not accurate enough, our framework provides a novel way to automatically get the importance ratio based on our proposed subjective test methodology and results, which is much easier and more accurate than previous assignment methods.

3.3 Bitrate Model

In this section, we develop a bitrate model which estimates the encoding bitrate of a view at a given time from the graphics rendering settings in this view at that time.

Several techniques have been proposed to model the bitrate of encoded video as a function of the video encoding parameter quantization step q and the video frame rate t. For example, in [48], Ma et al. proposed Equation (3.24)

$$R(q,t) = R_{\max}\left(\frac{q}{q_{\min}}\right)^{-a}\left(\frac{t}{t_{\max}}\right)^{b} \tag{3.24}$$

in which $q_{\min}$ and $t_{\max}$ parameters are chosen based on the application and $R_{\max}$ is the actual bitrate when encoding a video at $q_{\min}$ and $t_{\max}$. Coefficients a and b are model parameters which depend on the content of the video. The authors in [48] further proposed

a method to estimate a and b based on content features shown in Equations (3.25) and (3.26).

$$[abR_{\max}]^T = B[1\mu_{FD}\mu_{MVM}\frac{\mu_{MVM}}{\sigma_{MDA}}]^T \tag{3.25}$$

$$B = \begin{bmatrix} 1.1406 & -0.0330 & -0.0611 & 0.1408 \\ 0.4462 & 0.0112 & 0.0680 & -0.0667 \\ 0.1416 & -0.0008 & -0.0001 & -0.0036 \end{bmatrix} \tag{3.26}$$

in which μ_{FD} represents mean of frame difference, μ_{MVM} stands for mean of motion vector magnitude and σ_{MDA} means standard deviation of motion direction activity.

Although reported in [48, 70] that this model is of high accuracy, it is based on a data set containing videos that are natural scene videos and the resolutions are CIF (352×288) rather than the situation in our CMG(3D) application where the view is generated by computer instead of camera in the real world and the video resolution used is much higher: 720p (1280×720). Moreover, this work [48, 70] only considers video encoding parameters but do not consider any graphics rendering settings. Thus, the model Equations (3.24)-(3.26), especially the model parameters, may not be accurate enough in our CMG(3D) application. Therefore, in this work, we extend their work by:

A. Conduct experiments using CMG(3D) videos with 720p resolution to validate the model equations.

B. Conduct additional experiments to verify the B matrix in Equation (3.26).

C. Conduct additional experiments to predict μ_{FD}, μ_{MVM} and σ_{MDA} from the graphics rendering settings so that the model not only considers video content features but also graphics rendering settings.

In the following, we use x264 [57] as the video encoder. The version of x264 that we use is r2230. Table 3.5 lists the encoding parameters we use in our experiment. In addition, we set $QP)\min = 25$ and $t_{\max} = 60$. By using the H.264/AVC standard

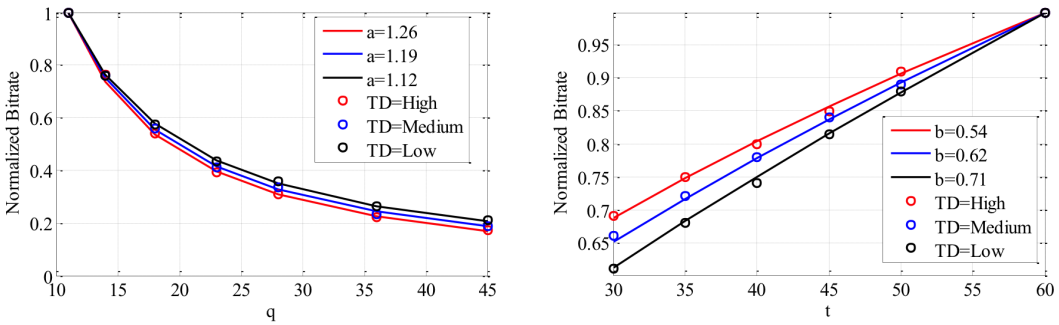
definition, $q = 2^{(}(QP-4)/6)$, $q_{\min} = 11$.

Table 3.5. x264 encoding parameters

Encoding Parameters	Value	Encoding Parameters	Value
Rate Control Method	X264_RC_CQP	Periodic Intra Refresh	1
Profile	High	QP	25
Level of IDC	0	Number of Frames per second	30
Sub Pixel Refine Value	2	Enable CABAC	0
Enable Motion Estimation for Chroma	0	Enable PSNR	1

3.3.1 Model Equation Validation

In this subsection, we introduce how we perform experiments to validate the model Equation (3.24).



(a) **(b)**

Figure 3.10. (a) left, Validation of model equation with q (b) right, Validation of model equation with t

In order to validate this model, first we fix $t = 60$ and capture 3 videos using our CMG(3D) system with different texture detail settings and encode them using different QP settings including 25, 27, 29, 31, 33, 35 and 37 (the corresponding q values are 11, 14, 18, 23, 28, 36 and 45). For each video, we encode it by using x264 software with parameters shown in Table 3.5 and record the bitrate under each q value. We calculate

the normalized bitrate $R(q)/R_{\max}$. Figure 3.10a shows the results. x-axis of the fig is q which ranges from 11 to 45 and y-axis is the normalized bitrate. The results of each video are represented by a specific color. Besides the bitrates shown as circles for the 3 videos with different texture details, we also plot a line for each video to represent the model equation.

The parameter a is obtained by minimizing the squared error between the model predicted and measured rates for each video. From Figure 3.10b , we can conclude that q and normalized bitrate follows Equation (3.24) proposed by [48].

Next, we fix $q = 11$ and change the framerate when we capture the game video. The framerate values we choose are 30, 35, 40, 45, 50 and 60. We capture 18 videos in total for each framerate setting with 3 texture detail settings. Again, for each video, we encode it by using x264 software with parameters shown in Table 3.5 and record the bitrate under each t value. We calculate normalized bitrate $R(t)/R_{\max}$. Figure 3.10b shows the results. X-axis of the figure is t which ranges from 30 to 60 and y-axis is the normalized bitrate. The results of each video are represented by a specific color. Besides the bitrates shown as circles for the 3 videos with different texture details, we also plot a line for each video to represent the model equation. The parameter b is obtained by minimizing the squared error between the model predicted and measured rates for each video. From Figure 3.10b , we can conclude that t and normalized bitrate also follows an exponential relationship so that the model equation proposed by [48] is still correct.

From the above, we conclude that the video bitrate model proposed by [48] still applies to a specific category of video types that is gaming video and also in a different resolution.

3.3.2 Model Parameter Validation

After validating that the model equations are accurate, we continue to validate whether the parameter matrix B proposed in previous work is accurate enough.

To achieve the above, we use the video data mentioned in the above subsection and divide them into two parts. We use 60% of the data to train the generalized linear predictor proposed by [70] and use 40% of the data to validate. The results we get for the parameter matrix B is the following:

$$B = \begin{bmatrix} 1.031 & -0.0812 & -0.0823 & 0.213 \\ 0.443 & 0.032 & 0.1870 & -0.0761 \\ 1.387 & -0.0042 & -0.0003 & -0.0043 \end{bmatrix} \tag{3.27}$$

Note that the mean of sum of absolute difference caused by using the B matrix in Equation (3.27) is 18.25kbps while it is 170.3kbps by using the B matrix proposed by [48] (Equation (3.26)). Thus, our conclusion is that the B matrix derived above (Equation (3.27)) is more accurate for gaming videos with 720p resolution than originally proposed, and hence we will use it in our work.

3.3.3 Relationship between Content Feature and Graphics Rendering Settings

Since in this work we assume there will be no video impairment due to encoding, we fix the values for the quantization step size q and the frame rate t. Thus, the only variables in the model equation will be content features (μ_{FD}, μ_{MVM} and σ_{MDA}). In the following, we perform several experiments to derive the relationship between rendering settings of each object and content features (μ_{FD}, μ_{MVM} and σ_{MDA}). Intuitively, suppose

we have s objects in total, then for example the function of μ_{FD} will become the following:

$$\mu_{FD} = F\left(TD_1, TD_2, \ldots, TD_s\right) \tag{3.28}$$

in which TD_i represents the rendering setting for object i.

Because the number of objects is changing all the time so that the number of the variables in the above function will be dynamic, it is difficult to derive the relationship directly. Hence, instead of using the rendering setting of every single object as one variable, we use the percentage of pixels with certain rendering setting as the variable. (Thus the number of variables will be fixed.) For example, for the game Broadsides, it has four settings, High texture detail, Medium texture detail, Low texture detail and not rendered at all. We propose the function to be the following:

$$\mu_{FD} = F\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k, \sum_{TD_{L\_l}=N} p_l\right) \tag{3.29}$$

The first variable $\sum_{TD_{L\_i}=H} p_i$ indicates the total percentage of pixels of all the objects that are rendered as High texture detail, the second for Medium texture detail, the third for Low texture detail and the fourth for those that are not rendered. Moreover, since the sum of the percentages of the above four conditions will be 100%, the above function can be further simplified to include only three variables. We remove the variable indicating the total percentage of pixels of all the objects that are not rendered. So that the function becomes:

$$\mu_{FD} = F\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right) \tag{3.30}$$

However, relating the content features to only the rendering settings studied in this chapter only may not be appropriate as the content features are also related to the

temporal and spatial complexity of the image, lighting and other rendering settings. On the other hand, content features can actually be calculated after one frame is encoded and our purpose is to predict what the bitrate will be if we substitute the rendering settings for an object or multiple objects. Thus, we propose to study the following relationship which will be more appropriate and more accurate.

$$\frac{\mu_{FD}\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right)}{\mu_{FD}(1,0,0)} = F\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right) \tag{3.31}$$

By introducing a division operation, the influence of other settings to the content features are removed automatically so that we can focus on the influence of the rendering settings studied in this chapter. Also the denominator in the equation represent the case when all the objects are rendered with High texture detail which is the default setting of the game and can be studied separately.

Similar to the methods used in [48, 67, 70], we develop a generalized linear predictor showing in Equations (3.32)-(3.34) to predict the relationship.

$$\frac{\mu_{MVM}\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right)}{\mu_{MVM}(1,0,0)} = H_1\left[1, \sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right]^T \tag{3.32}$$

$$\frac{\sigma_{MDA}\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right)}{\sigma_{MDA}(1,0,0)} = H_2\left[1, \sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right]^T \tag{3.33}$$

$$\frac{\mu_{FD}\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right)}{\mu_{FD}(1,0,0)} = H_3\left[1, \sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_j}=M} p_j, \sum_{TD_{L\_k}=L} p_k\right]^T \tag{3.34}$$

We then capture 60 video clips from playing the game Broadsides and 40 video clips from playing the game Planeshift. We divide the video clips into several groups, with each group containing playing the game in the same game scene, performing the same tasks with the same time period, but with different rendering settings of each object. Within each group of videos, we first set the texture details of all objects to be High and calculate μ_{FD}, μ_{MVM} and σ_{MDA}. Secondly, we change texture details of some objects to be Medium, or Low or not render them. We collect the bitrate associated with each video clip and perform generalized linear prediction. We divide the whole dataset into two parts, with 60% of the data to train the model and use 40% of the data to test. The results are shown in the following:

$$H1 = \begin{bmatrix} 0.534 & 0.637 & 0.436 & 0.562 \end{bmatrix} \tag{3.35}$$

$$H2 = \begin{bmatrix} 0.992 & 0.045 & 0.063 & 0.076 \end{bmatrix} \tag{3.36}$$

$$H3 = \begin{bmatrix} 1 & 0.88 & 0.48 & 0.25 \end{bmatrix} \tag{3.37}$$

Thus, Equations (3.24),(3.25),(3.27),(3.32),(3.33),(3.34),(3.35),(3.36),(3.37) complete the proposed bitrate model. We validate the accuracy of the model by conducting experiments using another group of video clips captured with random scenes. Figure 3.11 shows the validation results. The correlation is 0.9923, which proves the accuracy of the model.

3.4 Adaptation Algorithm

In the above sections, we have proposed 1) a user experience model which models cloud mobile 3D display gaming user experience as a function of rendering settings of each object and 2) a bitrate model which estimates video bitrate needed to encode the

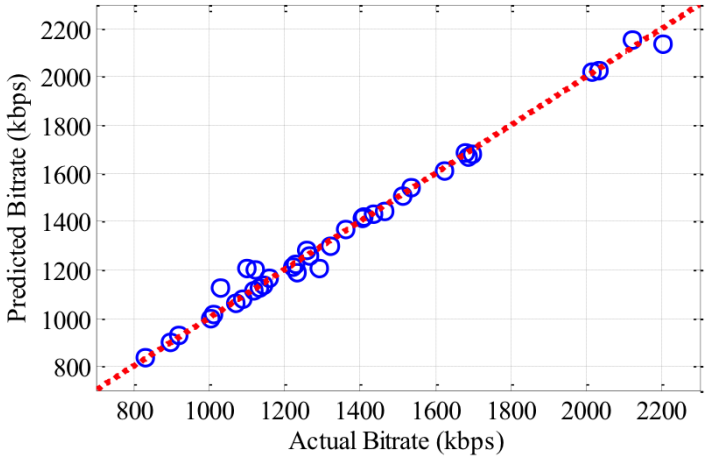


Figure 3.11. Validation results of the proposed bitrate model

Given:
1) Network bandwidth BW
2) Texture Detail bound $TD_{\min}$ and $TD_{\max}$
3) Percentage of pixels for each object p_i
4) Importance weight ratio c
5) Current content features μ_{FD}, μ_{MVM} and σ_{MDA}

Find:
The optimal graphics rendering factors, TDL_i and TDR_i, for each object i to minimize I_R

$$I^{opt} = \text{minimize}\,(I_R) = \text{minimize}\,(\min\,(I_S + I_{NS} - pI_S I_{NS}, 100)) \tag{3.38}$$

s.t.

$$\sum_{i=1}^{K_1} w_{KO} p_i + \sum_{j=1}^{K_2} w_{GO} p_j = 1, \frac{w_{KO}}{w_{GO}} = c \tag{3.39}$$

$$TD_{\min} \leq TD_{L\_i} \leq TD_{R\_i} \leq TD_{\max} \tag{3.40}$$

$$\begin{pmatrix} R_L\left(\sum_{TD_{L\_i}=H} p_i, \sum_{TD_{L\_i}=M} p_i, \sum_{TD_{L\_k}=L} p_k\right) + \\ R_R\left(\sum_{TD_{R\_i}=H} p_i, \sum_{TD_{R\_i}=M} p_i, \sum_{TD_{R\_k}=L} p_k\right) \end{pmatrix} \leq BW \tag{3.41}$$

Figure 3.12. Problem formulation

rendered video as a function of the rendering settings of each object. In this section, we combine these two models so that by selecting proper rendering settings of each object in each view we can find an optimal solution for maximizing user experience given a current network bandwidth constraint. Figure 3.12 shows the problem formulation. Because according to Equation (3.1) and (3.2), maximizing CMG(3D)-UE is equal to minimizing I, we set optimization objective to be minimizing I. $TD_{\min}$ and $TD_{\max}$ are the minimum and maximum boundaries of texture details being used for a game. So if the rendering engine uses texture detail values of High, Medium, Low, and we assign 0 to be High, 1 to be Medium and 2 to be Low, then $TD_{\min} = 0$, $TD_{\max} = 2$. Considering we also have an option to not render the object, we assign $TD = 3$ in this case. Thus $TD_{\min} = 0$, $TD_{\max} = 3$. The percentage of pixels object i occupies, p_i, can be obtained through game information and coefficient c has already been studied in the previous section. In addition, note that in Equation (3.41) we are using H.264/AVC video encoder to encode two views separately, thus the total bitrate equals the sum of the bitrate of two views. If we substitute the video encoder with H.264/MVC or video+depth encoder, the expression of total bitrate should be only one term.

In order to solve the above problem, we propose an Asymmetric and Selective Object Rendering Adaptation (ASORA) algorithm which runs periodically (in this chapter we set the time period to be 1 second) so that the rendering settings could be changed dynamically depending on changing network conditions. In detail, at the end of each time period, we will obtain the inputs for the ASORA algorithm: 1) current network bandwidth BW which is obtained through an active networking probing method we proposed in our previous work; 2) percentage of pixels for each object p_i and current content features (μ_{FD}, μ_{MVM} and σ_{MDA}). We use the p_i value and the content features in the current time period to predict the bitrate consumption in the next time period with different rendering settings. The output of the algorithm will be the optimal rendering

setting for each object in the next time period.

Intuitively, our problem is very similar to a Multi Choice 0/1 knapsack problem (MCKP) [72]. The definition of the MCKP is the following: Given a knapsack of capacity C and n items with associated values $v_1, v_2, \ldots, v_n$ and weights $w_1, w_2, \ldots, w_n$ and belonging to a set of k mutually exclusive classes $N_1, N_2, \ldots, N_k$ the MCKP is to find the subset of items consisting of exactly one item from each class that maximize total value without exceeding knapsack capacity. If we associate the impairment contributed by each object with a value in MCKP, video bitrate with weight and bandwidth with capacity, the two problems looks very similar. However, the difference of our problem is that the contribution to the impairment and bitrate of each object is not additive. Thus, we cannot use dynamical programming algorithm (which applies to MCKP) to solve our problem. In addition, our optimization problem can be categorized as a discrete variable, non-linear object function with non-linear constraint function problem which can be optimally solved using optimization solvers, such as CPLEX [75]. However, using CPLEX gives optimal solutions at the expense of exponential computation complexity. Thus, we propose the following algorithm which is both efficient and accurate.

The detail of the algorithm is the following. Considering the fact that the algorithm needs to be executed in as little time as possible as CMG(3D) application has a real time execution requirement, we propose ASORA which is an approximation algorithm based on greedy approach with pruning [73], and is shown in the following pseudo code. The underlying principle of the algorithm is as follows: initially we set all the objects to be not rendered in both left and right views. Then we keep adjusting the rendering setting of objects using a while loop, as long as the total bit rate does not exceed the bit rate budget (BW). During each iteration, the algorithm will firstly iterate over all the objects and for each object i, the algorithm computes the possible degradation in its rendering impairment (ΔI_i) and the possible increase in the consumed bandwidth (ΔBW_i), if we

set its rendering setting in one view to be one level higher. Among all the objects, the algorithm will choose the one with the highest ratio of $\Delta I_i/\Delta BW_i$. The algorithm will stop when 1) it reaches the maximum number of iteration bound or 2) there is no more bandwidth available or 3) all objects set texture detail to be High. The proposed ASORA algorithm has a run time of less than 12ms on a computer with a dual-core i7 processor, and thus can meet the real time execution requirement.

3.5 Experimental Results

In this section, we report on experiments conducted using a commercial cloud service to verify the performance improvement by applying the proposed ASORA technique over the other existing methods. We use the same testbed as shown in Figure 3.2, except we implement our CMG(3D) system, including the ASORA algorithm, on Amazon Web Service (AWS) servers. For the Amazon cloud server, the CPU is Intel Xeon E5-2670 @2.60GHz with 15GB memory and the GPU has 1536 CUDA cores and 4GB of video memory. The operating system is Windows Server 2008 R2 SP1.

We firstly collected real 4G-LTE network traces by using network bandwidth testing software Speedtest.net [74] to record the bandwidth around UCSD campus. Figures 3.13 and 3.16 show two sample LTE traces collected during lunch time and dinner time (both are peak usage hours of mobile network at university), which are then emulated using the network emulator in our testbed. In addition, for comparison reasons, we also implemented two other algorithms called ARA from the last chapter and JREA from [51]. Basically, ARA enables the game to set two different texture details for left view and right view, but all the objects in the same view will have the same texture detail settings. ARA also enables view distance settings which let the game not render the objects whose distance to the virtual camera in the world is greater than the threshold (value of view distance). JREA is an adaptation algorithm developed for 2D

Algorithm 2: Asymmetric and Selective Object Rendering Adaptation (ASORA)

Input :
1) Network bandwidth BW
2) Texture Detail bound $TD_{\min}$ and $TD_{\max}$
3) Percentage of pixels for each object p_i
4) Content features μ_{FD}, μ_{MVM} and σ_{MDA}
5) Importance weight ratio c
Output: I^{OPT}, ${TD_{L\_i}}^{OPT}$, ${TD_{R\_i}}^{OPT}$
Step 1: For each object i, set both ${TD_{L\_i}}^{OPT}$ and ${TD_{R\_i}}^{OPT}$ to be $TD_{\max}$, calculate the current bandwidth needed BW_{cur} and I_{cur}

$$BWpre = BWcur$$

$$Ipre = Icur$$

Step 2:
while $(BW - BWcur > 0)$ *&& (round < round_limit) &&* $!(all\ TD == 0)$ **do**
  for $i = 1; i < K; i++$ **do**
    $max({TD_{L\_i}}^{OPT}, {TD_{R\_i}}^{OPT}) = max({TD_{L\_i}}^{OPT}, {TD_{R\_i}}^{OPT}) - 1$
    calculate BW_{cur} and I_{cur}
    $\Delta BW_i = BW_{cur} - BW_{pre}$
    $\Delta I_i = I_{pre} - I_{cur}$
    $\max({TD_{L\_i}}^{OPT}, {TD_{R\_i}}^{OPT}) = \max({TD_{L\_i}}^{OPT}, {TD_{R\_i}}^{OPT}) + 1$
  end
  Find i which has the maximum value of $\Delta I_i / \Delta BW_i$
  Step 3: Set the change of TD setting of object i

$$\max({TD_{L\_i}}^{OPT}, {TD_{R\_i}}^{OPT}) = \max({TD_{L\_i}}^{OPT}, {TD_{R\_i}}^{OPT}) - 1$$

end
Step 4: **Return** I^{OPT}, ${TD_{L\_i}}^{OPT}$, ${TD_{R\_i}}^{OPT}$

CMG applications. The basic idea for this technique is that it pre-defines several groups of parameter combinations and assigns them into different levels. The algorithm will choose to go up a level or go down a level at a time when the network conditions changes. We extend the framework of it for CMG(3D), but only let the texture detail settings to be the same for all the objects in both of the views. We evaluated the performance of the algorithms on both Broadsides and Planeshift.

Figures 3.13 and 3.16 show the network bandwidth profiles. The average bandwidth for network bandwidth profile 1 is 3330 kbps and it is 3029 kbps for network bandwidth profile 2. Thus, network bandwidth profile 2 is more challenging.

Under the bandwidth constraint, the three algorithms will compute the best texture detail settings they support and set these parameters in the game which will result in different video bitrate consumption and different impairment for users. Figures 3.14a 3.15a 3.17a 3.18a show the bitrates resulting from using all three algorithms. Figures 3.14b 3.15b 3.17b 3.18b show the resulting rendering impairment I_R of the three algorithms. Figures 3.14c 3.15c 3.17c 3.18c show the resulting CMG(3D)-UE scores of the three algorithms computed using the proposed user experience model. The statistical results are all shown in Table 3.6. Our experimental results show that most of the time, ASORA performs significantly better than the other two algorithms. For example, in Figure 3.15c, at time= 200s, we can see that using ASORA enables achieving significantly better user experience than using the other algorithms. By using ASORA, the CMG(3D)-UE score is 4 which means negligible visual impairment, while by using ARA and JREA, the resulting CMG(3D)-UE score is 1 which means unacceptable visual impairment that will cause the user to quit the game. Further, if we compare the results using two different network bandwidth profiles, we can see that the performance gain of ASORA over the other two algorithms increases with more challenging network conditions. Moreover, if we compare the results using two games, we can observe that ASORA performs better

using Planeshift than Broadsides, the main reason is the following. From Table 3.3, it shows the average importance ratio of Planeshift is higher than that of Broadsides, thus, by transferring more low quality rendering settings to more unimportant objects from important objects while preserving the video bitrate, Planeshift can get much more user experience gain than Broadsides. Overall we can see that ASORA performs much better than the other two algorithms. Especially from the statistical results we can see that ASORA performs 49.4%-70.3% better than ARA and 90.8%-124.8% better than JREA in terms of CMG(3D)-UE score.

The above results prove that the proposed algorithm is much better than existing approaches and can be applied in different network conditions with different genres of games.

Table 3.6. Statistical results of the experiment showing rendering impairment I_R and overall user experience achieved CMG(3D)-UE.

Network Profile	Game	I_R			CMG(3D)-UE		
		ASORA	ARA	JREA	ASORA	ARA	JREA
Profile1	Broadsides	13.59	50.69	66.16	4.14	2.77	2.17
Profile1	Planeshift	8.96	51.37	67.71	4.26	2.79	2.17
Profile2	Broadsides	35.07	70.43	83.36	3.32	2.01	1.52
Profile2	Planeshift	31.13	71.59	84.86	3.44	2.02	1.53

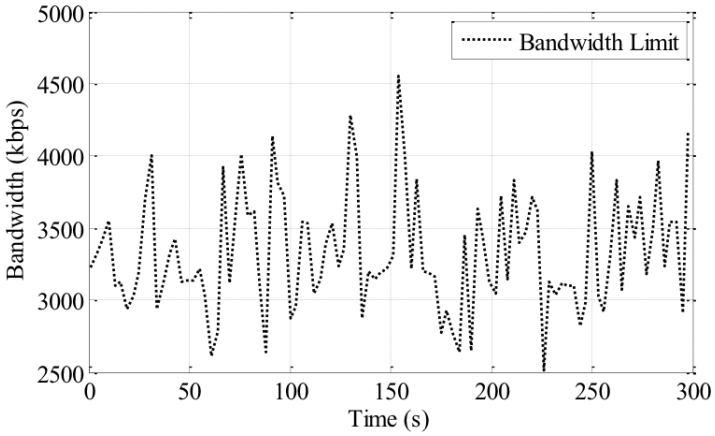
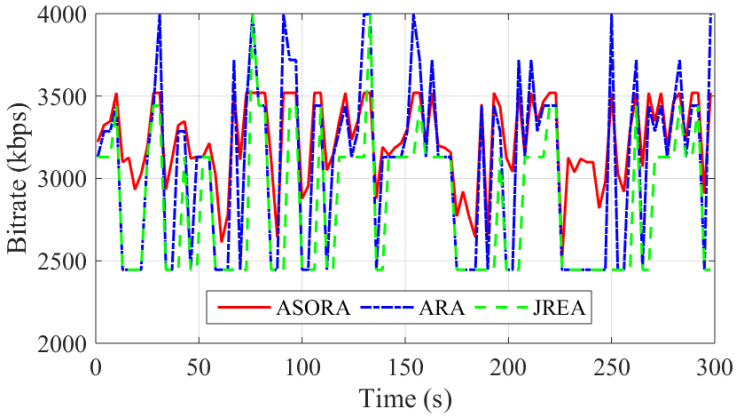
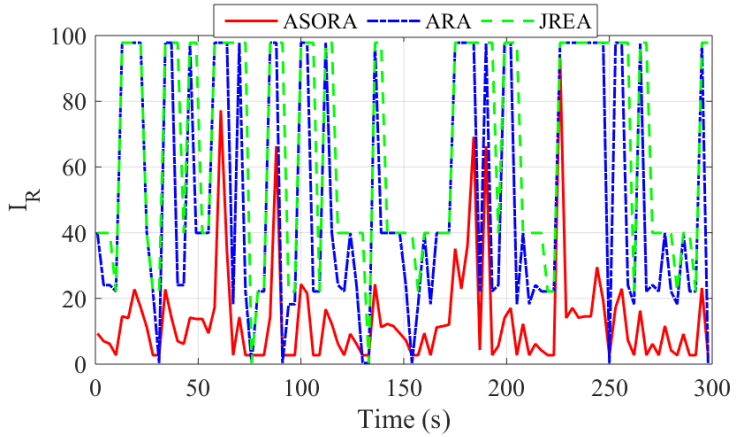


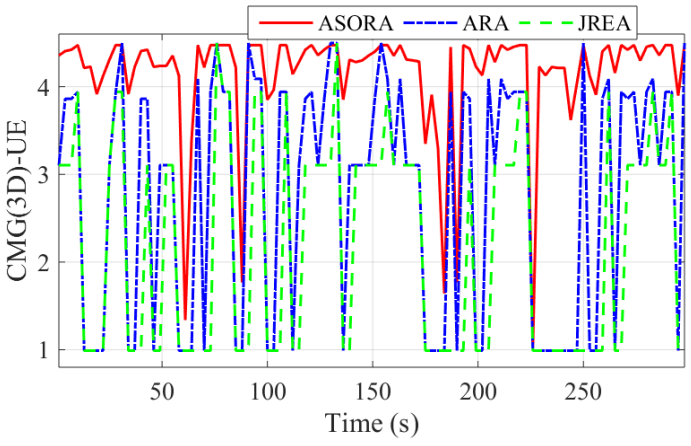
Figure 3.13. Network bandwidth profile 1



(a)

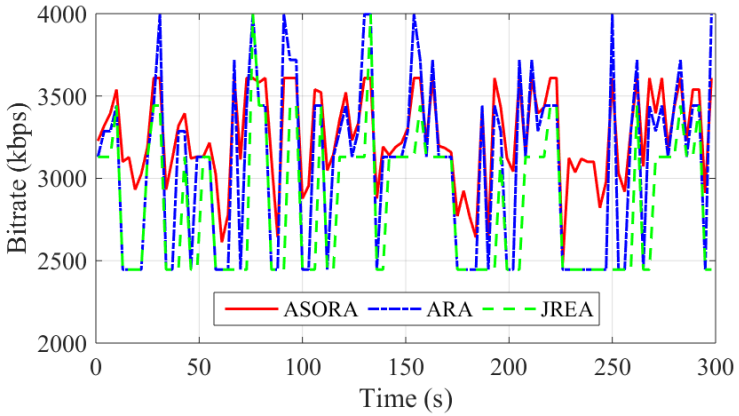


(b)

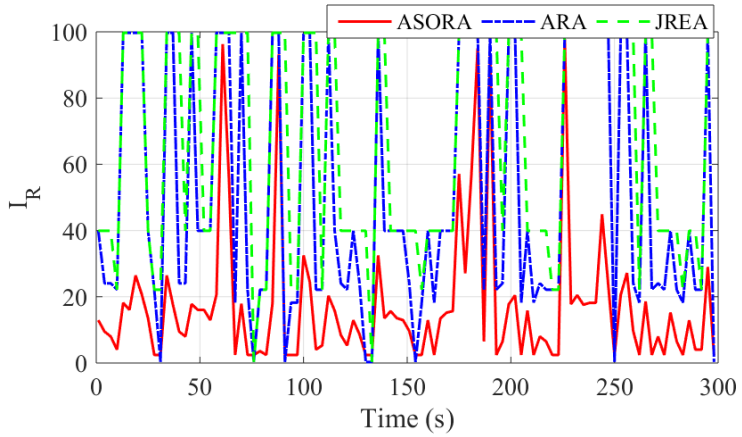


(c)

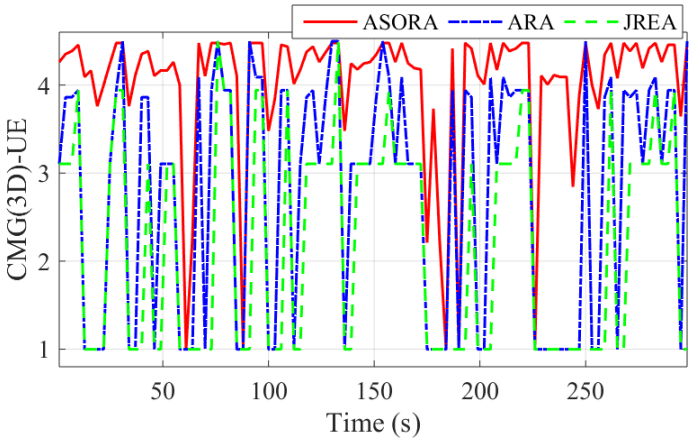
Figure 3.14. Results of Broadsides using network bandwidth profile 1 (a) top, Bitrate of three algorithms (b) middle, IR of three algorithms (c) bottom, CMG(3D)-UE of three algorithms



(a)



(b)



(c)

Figure 3.15. Results of Planeshift using network bandwidth profile 1 (a) top, Bitrate of three algorithms (b) middle, IR of three algorithms (c) bottom, CMG(3D)-UE of three algorithms

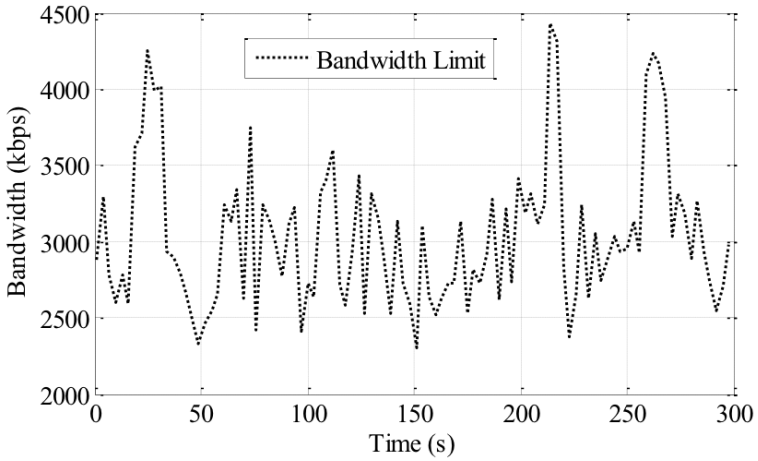


Figure 3.16. Network bandwidth profile 2

3.6 Conclusion

The main contributions of this chapter are the following:

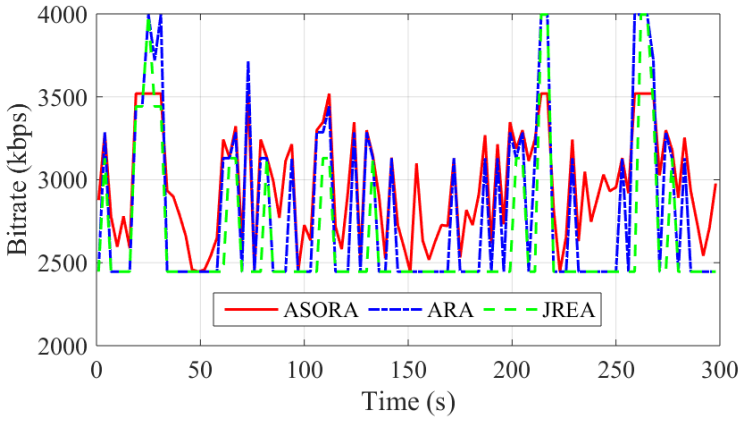
1) We performed extensive subjective tests to derive a general user experience model for cloud mobile 3D display gaming considering both asymmetric object rendering and selective object rendering.

2) We derived a video bitrate model to relate the video with the changes of different rendering settings.

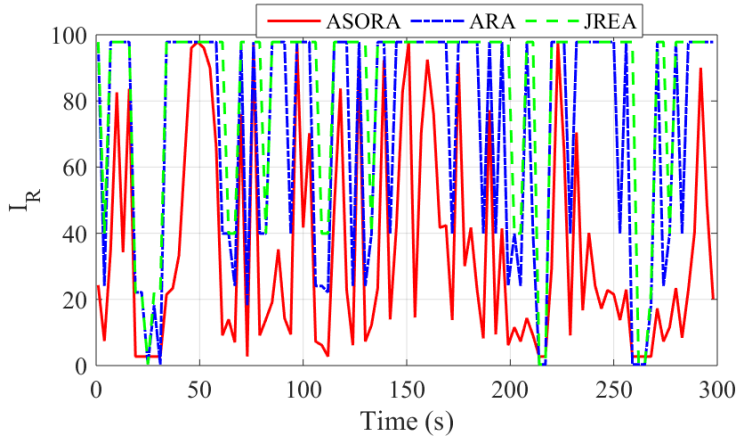
3) By making use of the above two models, we developed a novel adaptation algorithm called ASORA that is able to automatically decide the optimal rendering settings for objects in left and right views to ensure the best user experience given the network conditions.

4) By conducting experiments using real 4G-LTE network profiles and commercial cloud service and comparing with existing methods, we demonstrate the significant improvement in user experience when the proposed ASORA algorithm is applied.

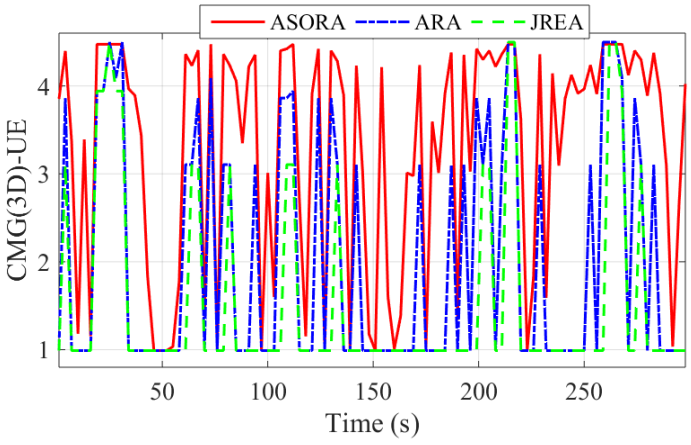
In the next chapter, we will not only consider asymmetric graphics rendering, but apply it together with asymmetric video encoding to further improve the user experience.



(a)

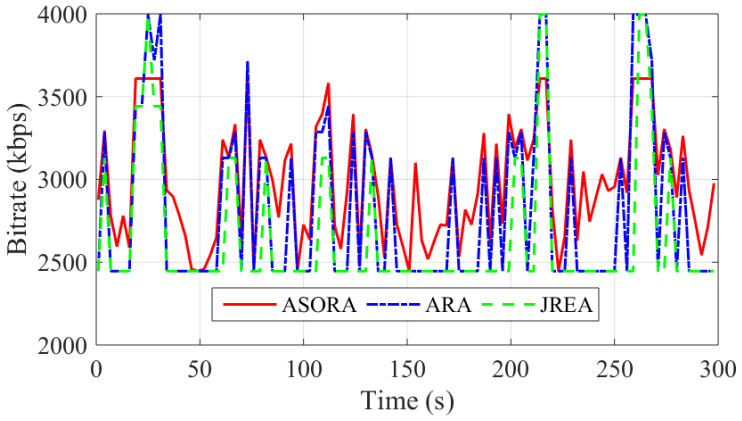


(b)

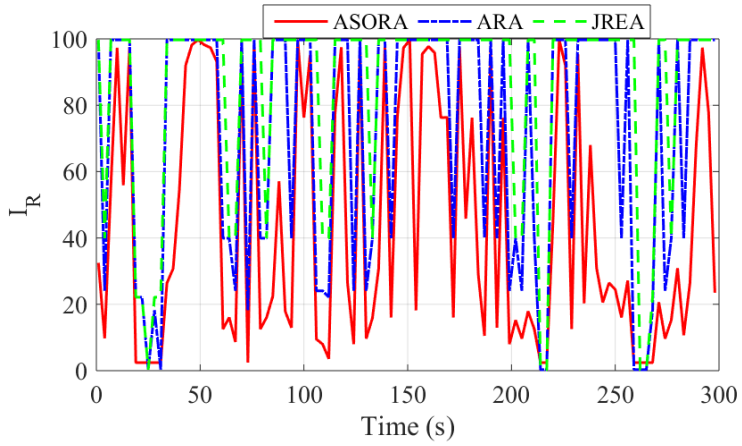


(c)

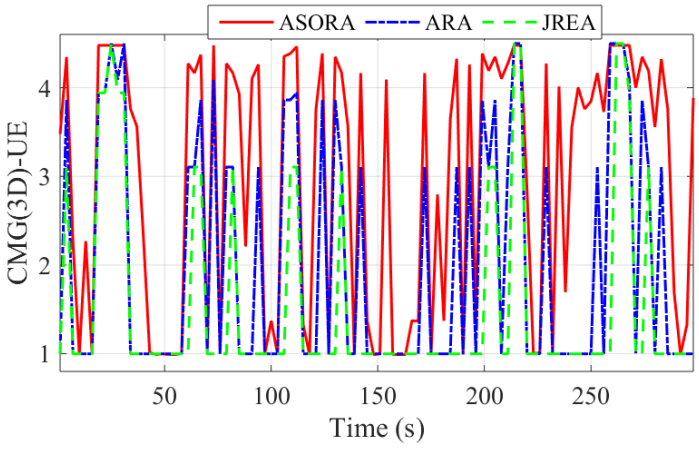
Figure 3.17. Results of Broadsides using network bandwidth profile 2 (a) top, Bitrate of three algorithms (b) middle, IR of three algorithms (c) bottom, CMG(3D)-UE of three algorithms



(a)



(b)



(c)

Figure 3.18. Results of Planeshift using network bandwidth profile 2 (a) top, Bitrate of three algorithms (b) middle, IR of three algorithms (c) bottom, CMG(3D)-UE of three algorithms

3.7 Acknowledgements

Chapter 3, in part, is from the material as it appears in proceedings of IEEE ICC 2015. Yao Lu; Yao Liu; Sujit Dey. and in Multimedia Tools and Applications 2016. Yao Lu; Yao Liu; Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Joint Asymmetric Video Encoding and Graphics Rendering

4.1 Introduction

In the last two chapters, asymmetric graphics rendering has been extensively studied. We proposed asymmetric texture detail. We also proposed selective rendering for each individual objects. The experimental results proved the effectiveness of the proposed technique. In this chapter, we further explore the idea of combining asymmetric graphics rendering and asymmetric video encoding together to further improve user experience. As is the work flow of the previous two chapters, we propose the idea, model the user experience, develop a bit rate model and design an algorithm to solve the optimization problem. Compared to what is proposed in the previous chapters, we have the following improvements. First, we extend the application from game only to support other virtual immersive applications. We term them as cloud mobile 3D virtual immersive applications (CMVIA(3D)). Second, we replace the H.264/AVC codec to be H.265/HEVC codec to improve the coding efficiency.

4.2 User experience model

In this section, we study and model user experience for CMVIA(3D) considering the joint effect of video encoding impairment and graphics rendering impairment, as opposed to our previous work which only takes into account the effects of graphics rendering impairment for CMG(3D). Because both the system architecture and the factors that influence user experience of CMVIA(3D) and CMG(3D) are very similar, we use the same Mean Opinion Score (MOS) criterion introduced in Chapter 2 as a measurement metric for modeling CMVIA(3D) user experience, as shown in equations (4.1) and (4.2):

$$\text{MOS} = 1 + 0.035R + 7 \times 10^{-6} R\,(R-60)\,(100-R) \tag{4.1}$$

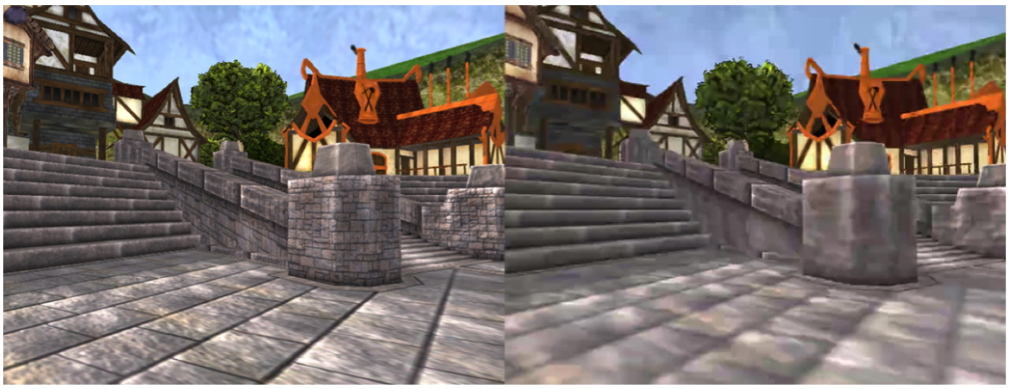
$$R = 100 - I\,(VE, GR) \tag{4.2}$$

In Equation (4.1), the MOS metric is formulated by a transmission rating factor R, which represents the overall user experience. R factor takes value between 0 and 100; higher R value means higher MOS and better user experience. In Equation (4.2), the equation $I(VE, GR)$ stands for the combined impairment caused by video encoding (VE) and graphics rendering (GR).

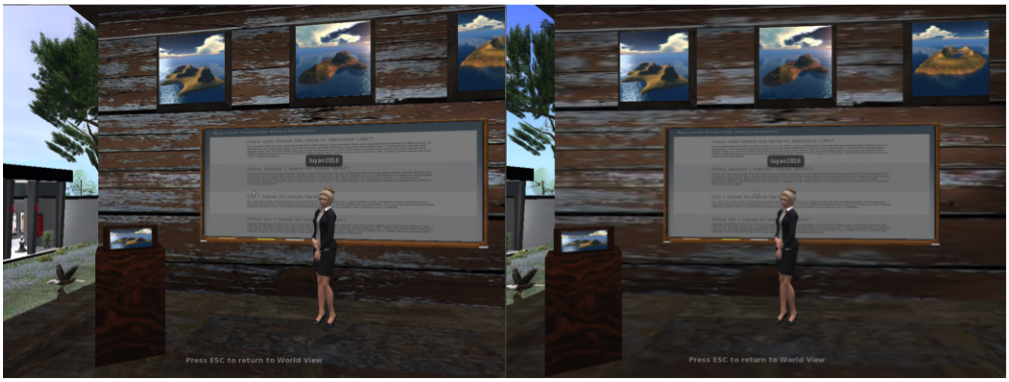
In the following, we describe how we conducted subjective tests to derive the impairment functions and how we validated them.

4.2.1 Subjective Test settings

Table 4.1 shows the specific video encoding settings and graphics rendering settings we want to include in our experiment. In detail, for video encoding settings, we fix the resolution to be 720p (960×720) for each view and frame rate to be 25fps in our experiment but change the quantization parameter (QP). QP that ranges from 0



(a)



(b)

(c)

Figure 4.1. Example snapshots for three applications (a) top, Cloud mobile 3D display gaming left view: High texture detail, $QP = 25$, right view: Medium texture detail, $QP = 35$ (b) middle, Cloud mobile 3D virtual classroom left view: High texture detail, $QP = 25$, right view: Medium texture detail, $QP = 27$ (c) bottom, Cloud mobile 3D virtual art gallery left view: High texture detail, $QP = 25$, right view: Medium texture detail, $QP = 25$

to 51 decides quantization level. Higher QP means lower quality. The other settings of x265 encoder are listed in Table 4.2. For graphics rendering settings, we study the effect of asymmetric texture detail. Texture detail defines the quality of the images on the surface of the objects. As is defined in Chapter 2, we define texture detail to be high when the game is using the original texture images, to be medium when the texture images are downsampled once and low when the texture images are downsampled twice. Figure 4.1 shows an example where left view is rendered with high quality and right view is rendered with medium quality. The viewer can observe some blurry effect from the right view.

Figure 4.2. Testbed for subjective experiments

Figure 4.2 shows the testbed used for the subjective tests. We use a 3D monitor with a laptop to substitute for 3D display of mobile devices because current available mobile 3D displays do not have as good quality as 3D monitors that may cause additional impairment which we want to avoid. The laptop is connected to an access point and the access point is connected to the game server. The selected applications which run upon the above framework are 1) an online open-source MMORPG game PlaneShift [57] 2) an virtual classroom application we developed based on SecondLife [76] and 3) an virtual art gallery application we also developed based on SecondLife. We then invited 25 students (17 male, 8 female; aged 18$\sim$26) to participate in our subjective experiments. Firstly, we asked the testers to sit before a 23 inch LG D2342 3D Monitor, and show them a 3D video as a training sequence before the real test starts to let the testers adjust their viewing angle. After that, we start these applications and manually set the video encoding

settings and graphics rendering settings according to Table 4.1 independently for each view. Once a combination of rendering and encoding factors is set, we ask the testers to play the game for 1 minute and evaluate the impairment according to the criterion listed in Table 4.3 at the end of each condition. During the whole experiment, the testers were asked to control the avatar to perform multiple tasks (including attacking an enemy in the gaming application, discussing with another student in the virtual classroom application, watch the paintings in the virtual art gallery application, etc.). Example snapshots of the three applications with specific graphics rendering and video encoding settings are shown in Figure 4.1.

4.2.2 Impairment Function Derivation

Considering the graphics rendering and video encoding settings used in our CMVIA(3D) platform and the two views, we formulate impairment function as:.

$$I(VE, GR) = I(TD_L, TD_R, QP_L, QP_R) \tag{4.3}$$

where TD means texture detail and QP indicates quantization parameter. The subscripts L and R represent left view and right view respectively. In order to study $I(VE, GR)$, we first keep one of the four parameters fixed to its best quality value during the test and see how the impairment I changes according to other three parameters.

Table 4.1. Experiment setting

Settings	Experiment Values
Texture Detail (Down Sample)	High(0) Medium(1) Low(2)
Quantization Parameter	25 27 29 31 33 35 37 39

Figures 4.3a - 4.3f show the average impairment values when we keep QP of one of the views to be 25 (almost no video encoding impairment on that view) but change QP of the other view and at the same time change the texture detail settings for both views.

Table 4.2. x265 encoding parameters

Encoding Parameters	**Value**	**Encoding Parameters**	**Value**
Rate Control	CQP	Number of Reference frames	3
Profile	Main	Number of B frames	0
Preset	Medium	Scenecut	40
Rc-lookahead	0	Period Intra Refresh	ON
Wavefront Parallel Processing	ON	Weighted P Frame	ON
AMP Partition	ON	Weighted B Frame	OFF
Motion Estimation Range	57	Enable PSNR	ON

Table 4.3. Impairment criterion

I	**Description**
0	No visual impairment
0-20	Minor visual impairment
20-40	Noticeable visual impairment
40-60	Clear visual impairment
60-100	Unacceptable visual impairment

Table 4.4. QP threshold $T(TD)$ for different texture detail settings

Application	H	M	L
Gaming	27	31	35
Virtual Classroom	27	31	35
Virtual Art Gallery	27	29	31

Table 4.5. Average I_{TD} scores for different texture detail combinations

Application	H-H	H-M	M-M	M-L	L-L
Gaming	0.2	8.35	12.3	19.14	24.5
Virtual Classroom	0	7.4	11.5	17.2	22.6
Virtual Art Gallery	0	11.2	15.7	24.3	32.3

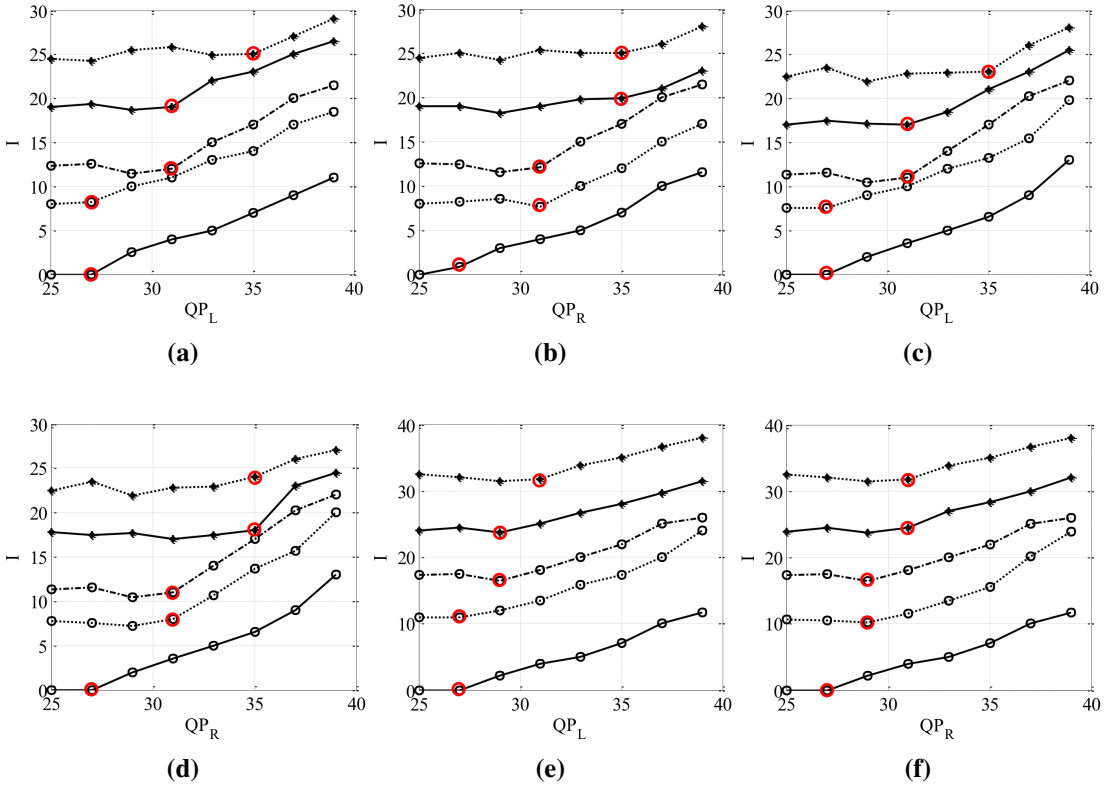


Figure 4.3. (a) top left, Relationship between I and QP_L under different texture detail combinations for gaming (b) bottom left, Relationship between I and QP_R under different texture detail combinations for gaming (c) top middle, Relationship between I and QP_L under different texture detail combinations for virtual classroom (d) bottom middle, Relationship between I and QP_R under different texture detail combinations for virtual classroom (e) top right, Relationship between I and QP_L under different texture detail combinations for virtual art gallery (f) bottom right, Relationship between I and QP_R under different texture detail combinations for virtual art gallery

From these six figures we can clearly observe that for each texture detail combination, the impairment values remain similar till the QP value exceeds some thresholds (showing as a red circle on the figures). For example, for curve L-L (both left and right views use low texture detail) in Figure 4.3a, the threshold is at 35 since the impairment I does not change until QP exceeds 35. Clearly the value of this QP threshold is related to the texture detail setting. Further, we found that this threshold was only related to the texture detail setting of the view whose QP is changing. For example, in Figure 4.3a, the threshold of M-M is the same as the threshold of M-L and that value is also the same as the threshold of H-M and M-M in Figure 4.3b. Thus, it means for a specific texture detail setting of one view, there is a corresponding QP threshold so that when QP is less than or equal to that threshold, the total impairment is not related to the QP value. In other words, the video encoding won't cause additional impairment besides the impairment caused by texture detail when the QP is below the threshold. According to the results from Figures 4.3a - 4.3f, we list the threshold and the corresponding impairment I_{TD} in Tables 4.4 and 4.5. Note that although the above relationship as well as all the models derived subsequently are general, the values of the parameters need to be derived for each specific application. From Tables 4.4 and 4.5 we can also identify that virtual art gallery application has much higher impairment due to texture detail and lower QP threshold than other two applications. The reason is that for this application, the user will pay continuous attention to the content of the virtual paintings so that they tend to be very strict about the details of the views. Thus, when there is some blurry effect of the views, it hurts user experience severely.

Considering the QP threshold, we propose to model I by two parts. The first one is the I_{TD} that is the impairment caused by texture detail only and the second part is I_A that is the additional impairment when QP is bigger than the threshold. Equation (4.4)

shows the relationship.

$$I(TD_L, TD_R, QP_L, QP_R) = I_{TD}(TD_L, TD_R) + I_A$$

$$I_A = \begin{cases} 0 & QP_L < T(TD_L) \\ & QP_R < T(TD_R) \\ f(QP_L - T(TD_L)) & QP_L \geq T(TD_L) \\ & QP_R < T(TD_R) \\ f(QP_R - T(TD_R)) & QP_L < T(TD_L) \\ & QP_R \geq T(TD_R) \\ g(QP_L - T(TD_L), QP_R - T(TD_R)) & QP_L \geq T(TD_L) \\ & QP_R \geq T(TD_R) \end{cases} \tag{4.4}$$

in which I_{TD} is the value form Table 4.5. The QP threshold $T(TD)$ is the value from Table 4.4.

From Figures 4.3a - 4.3f we observe that the impairment I would increase almost linearly with QP when QP exceeds the threshold $T(TD)$. Hence can conclude that f can be modeled as a linear function showing in Equation (4.5).

$$f(QP_L - T(TD_L)) = a(QP_L - T(TD_L)) + b \tag{4.5}$$

The parameters a and b for 3 applications are derived using linear regression technique and are listed in Table 4.6.

In order to derive g function, we change both QPs to be values that exceeds the $T(TD)$, plot the data points in a 3 dimensional figure and derive the relationship. Figure 4.4 shows an example for gaming application with texture detail combination set to be High-High. We tried different two dimensional equations to fit the data and we

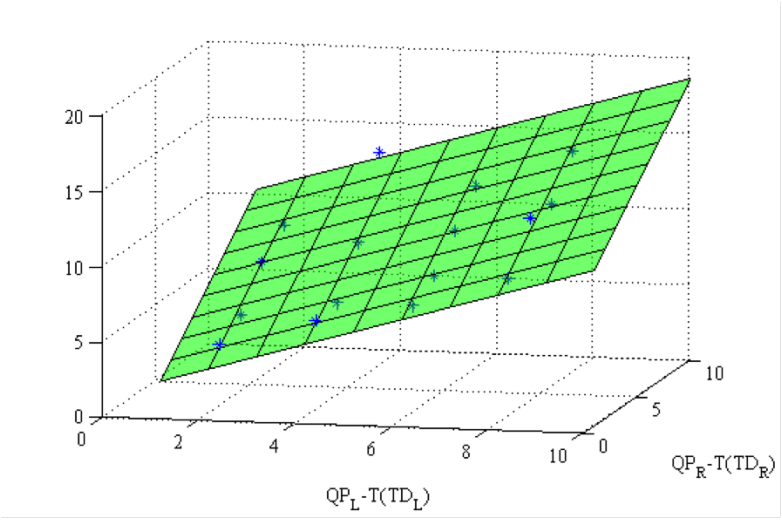


Figure 4.4. Regression result for g function for gaming application when texture detail combination is High-High

found that the bilinear equation can have the best regression results in terms Mean Square Error (MSE). Thus we model g function as is shown in Equation (4.6). The parameters a_1, a_2 and b_1 for 3 applications are shown in Table 4.7.

$$\begin{aligned} &g\left(QP_L - T\left(TD_L\right), QP_R - T\left(TD_R\right)\right) = \\ &a_1\left(QP_L - T\left(TD_L\right)\right) + a_2\left(QP_R - T\left(TD_R\right)\right) + b_1 \end{aligned} \tag{4.6}$$

Equations (4.3),(4.4),(4.5) and (4.6) complete our user experience model. We will validate it in the next subsection.

Table 4.6. Regression parameters of a and b for f function

Application	a			b		
	H	M	L	H	M	L
Gaming	0.9	0.8	1.1	0.1	0.3	0.2
Virtual Classroom	0.95	0.88	1.03	0.4	0.2	0.25
Virtual Art Gallery	0.88	0.75	1.21	0.3	0.23	0.14

Table 4.7. Regression parameters of a_1, a_2 and b_1 for g function

Application	H-H	H-M	M-M	M-L	L-L
	a_1				
Gaming	0.92	0.92	0.85	0.82	1.04
Virtual Classroom	0.98	0.96	0.91	0.90	1.13
Virtual Art Gallery	0.91	0.88	0.79	0.76	1.24
	a_2				
Gaming	0.92	0.83	0.85	1.01	1.04
Virtual Classroom	0.98	0.89	0.91	1.05	1.13
Virtual Art Gallery	0.91	0.72	0.79	1.23	1.24
	b_1				
Gaming	0.13	0.12	0.27	0.3	0.22
Virtual Classroom	0.41	0.43	0.22	0.23	0.25
Virtual Art Gallery	0.32	0.34	0.24	0.23	0.15

4.2.3 Model Validation

In order to validate the impairment model (Equations (4.3) $\sim$ (4.6)) derived in the previous subsections, we conducted another set of experiments with a new group of 16 participants (10 male, 6 female; aged 18$\sim$25), exploring the same applications. In this set of experiments, the texture detail and quantization parameters for both views are changed at the same time. Figures 4.5 shows the relationship between predicted impairment I computed by the derived impairment function (y-axis) and subjective impairment I given by human subjects (x-axis) for the 3 applications. In the figures, each data point represents one combination of graphics rendering settings and video encoding settings. We also plotted 95% confidence interval for each measurement as black lines in the figures to show the variety among different subjects. The correlation for the game application is 0.96 while it is 0.97 for virtual classroom application and 0.97 for virtual art gallery

application. The above results show the accuracy of the derived user experience model, and its applicability to different applications.

4.3 Bitrate Model

In TCP based cloud streaming applications, when video bitrate exceeds the network bandwidth, it will cause accumulated delay [59]. Therefore a bitrate model needs to be developed so that given a combination of parameters (TD_L, TD_R, QP_L, QP_R), the video bitrate can be accurately estimated. In this way, by controlling the parameter settings, the resulting video bitrate can be controlled to be below the network bandwidth and therefore avoid congestion.

Several techniques have been proposed to model the bitrate of encoded video as a function of the video encoding parameters. For example, in [48], Ma et al. proposed Equation (4.7) to model bit rate R using quantization step q and video frame rate t.

$$R(q,t) = R_{\max}\left(\frac{q}{q_{\min}}\right)^{-\alpha}\left(\frac{t}{t_{\max}}\right)^{\beta} \tag{4.7}$$

In Equation (4.7), coefficients $q_{\min}$ and $t_{\max}$ represent the minimum quantization step and the maximum frame rate, respectively, and are chosen based on the application; $R_{\max}$ indicates the maximum bitrate when encoding a video at $q_{\min}$ and $t_{\max}$; coefficients α and β are model parameters that depend on the content of the video. The authors in [48] further proposed a method to estimate α and β based on content features shown in Equation (4.8) and (4.9).

$$\begin{bmatrix}\alpha & \beta\end{bmatrix}^T = B\begin{bmatrix}1 & \mu_{FD} & \mu_{MVM} & \frac{\mu_{MVM}}{\sigma_{MDA}}\end{bmatrix}^T \tag{4.8}$$

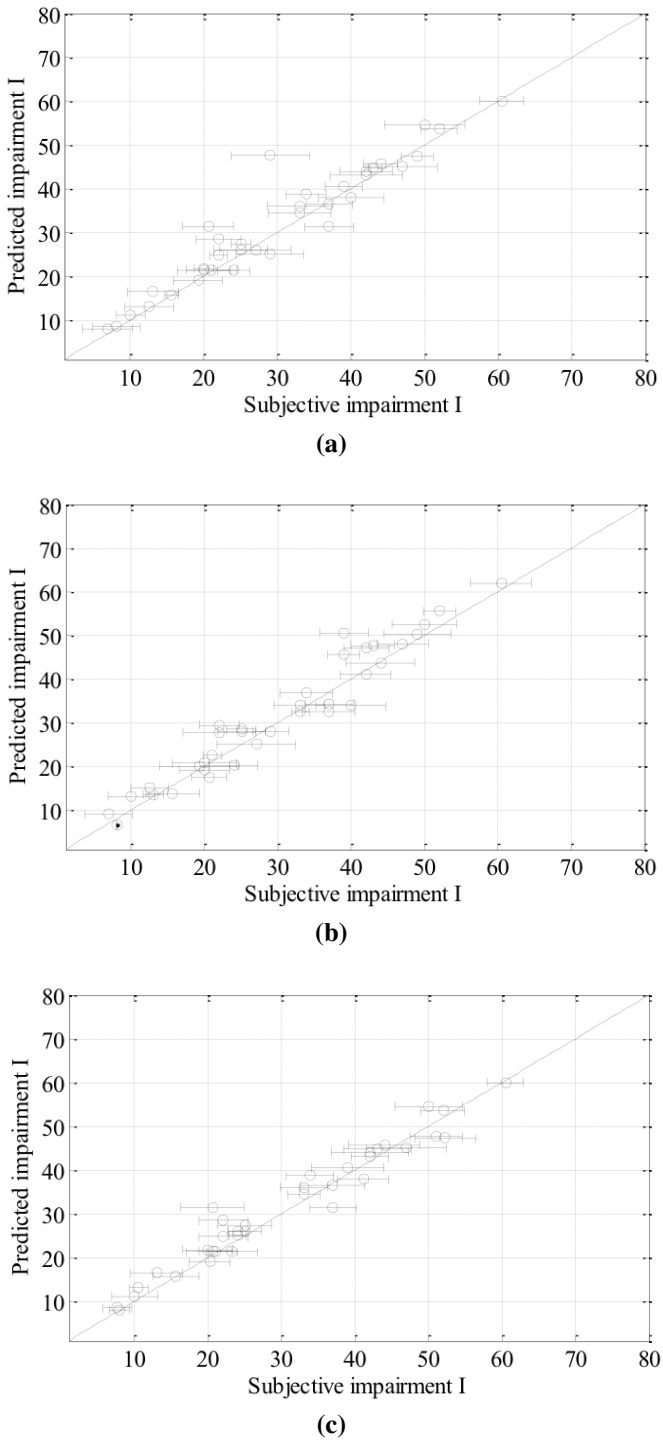


Figure 4.5. Validation of $I(VE, GR)$ (a) top, Results for gaming (b), middle Results for virtual classroom (c) bottom, Results for virtual art gallery

$$B = \begin{bmatrix} 1.1406 & -0.0330 & -0.0611 & 0.1408 \\ 0.4462 & 0.0112 & 0.0680 & -0.0667 \end{bmatrix} \tag{4.9}$$

in which μ_{FD} represents mean of frame difference, μ_{MVM} stands for mean of motion vector magnitude and σ_{MDA} means standard deviation of motion direction activity.

Though reported in [48] that this model is of high accuracy, it is based on a data set containing videos that are natural scene videos and the resolutions are CIF (352×288) rather than the situation in our CMVIA(3D) application where the view is generated by computer instead of camera in the real world and the video resolution for each view is 720p (960×720). Moreover, the video encoding standard we use is H.265/ HEVC instead of H.264/AVC in [48]. Also note that we need to consider both video encoding settings and graphics rendering settings in the bitrate model. Thus, the model Equations (4.7) $\sim$ (4.9), especially the model parameter may not be accurate enough in our case. Therefore, in this chapter we extend their work by:

1. Performing experiments using CMIVA(3D) videos with 720p resolution to validate the model equations.

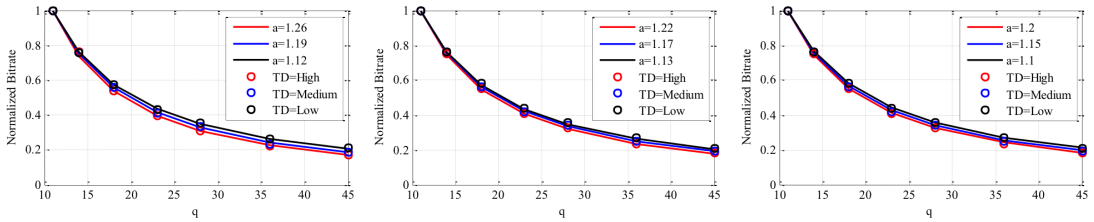
2. Performing additional experiments to adjust the model of parameter a (Equation (4.8),(4.9)) by incorporating graphics rendering setting with H.265/ HEVC video coding standard.

4.3.1 Model Equation Validation

In this subsection, we introduce how we perform experiments to validate the model equations. Firstly, because in this chapter, we do not evaluate the influence of framerate to user experience or bitrate, we will fix the framerate and set t to be $t_{\max}$ and thus we can simplify equation (4.7) to be:

$$R(q) = R_{\max}\left(\frac{q}{q_{\min}}\right)^{-\alpha} \tag{4.10}$$

In order to derive and validate this bitrate model, we captured 3 videos for each application using our CMVIA(3D) system with different texture detail settings and different QP settings from Table 4.1. Using the H.265/HEVC standard definition that $q = 2^{(QP-4)/6}$, the corresponding q values are 11, 14, 18, 23, 28, 36, and 45. For each video, we encode it by using x265 encoding library and record the bitrate under each q value. We set $R_{\max}$ to be the bitrate when encoding with $q_{\min}$ and calculate normalized bitrate $R(q)/R_{\max}$. Figures 4.6 show the results. X-axis of the figures is q which ranges from 11 to 45 and y-axis is the normalized bitrate. The results of each video in each figure are represented by a specific color. Besides the bit rates shown as circles for the 3 videos with different texture details, we also plot a line for each video to represent the model equation. The parameter α is obtained by minimizing the mean square error between the model predicted and measured rates for each video. From Figure 4.6, we can conclude the Equation (4.10) can model the bitrate of CMVIA(3D) videos using H.265/HEVC standard with high accuracy.



(a) **(b)** **(c)**

Figure 4.6. Validation of model equation (a) left, For gaming; (b) middle, For virtual classroom; (c) right, For virtual art gallery

4.3.2 Model Parameter Prediction

In this subsection, we discuss how we adjust the parameter model (Equation (4.8),(4.9)) proposed in [48] to cope with CMVIA(3D) application. As is reported in [48] that μ_{FD}, μ_{MVM} and μ_{MVM}/σ_{MDA} are the most related content features which influence parameter

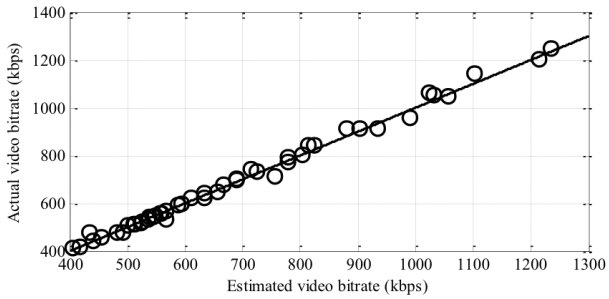


Figure 4.7. Validation of bitrate estimation

Given:
1) Video content features μ_{FD}, μ_{MVM} and σ_{MDA}
2) Network bandwidth limit BW
3) Texture Detail bound $TD_{\min}$ and $TD_{\max}$
4) Quantization Parameter bound $QP_{\min}$ and $QP_{\max}$
Find: The optimal TD_L, TD_R, QP_L, and QP_R to minimize impairment

$$I^{OPT} = \min I(VE, GR) = \min I(TD_L, TD_R, QP_L, QP_R)$$

s.t.

$$TD_{\min} \leq TD_L \leq TD_R \leq TD_{\max}$$

$$QP_{\min} \leq QP_L \leq QP_R \leq QP_{\max}$$

$$R_L(TD_L, QP_L) + R_R(TD_R, QP_R) \leq BW$$

Figure 4.8. Problem formulation

a. However, as is shown in Figure 4.5, the parameter a will vary for different texture detail settings. Thus we combine texture detail, TD, with the content features including σ_{FD}, μ_{MVM}/σ_{MVM}, etc. proposed in [48] as the input parameters for predicting a. Further, we captured 36 30-second long video clips with different texture detail settings and for different applications performing different tasks. We use the same generalized linear predictor with leave-one-out cross-validation error method reported in [48] to derive and validate the equations, which are shown in Equation (4.11),(4.12).

$$\alpha = B\begin{bmatrix} 1 & \mu_{FD} & \mu_{MVM} & \frac{\mu_{MVM}}{\sigma_{MDA}} & TD \end{bmatrix}^T \tag{4.11}$$

$$B = \begin{bmatrix} 1.13 & -0.076 & -0.042 & 0.00132 & 0.31 \end{bmatrix} \tag{4.12}$$

Our results show that using μ_{FD}, μ_{MVM}, μ_{MVM}/σ_{MDA} and TD is sufficient to predict the model parameter accurately. Figure 4.7 shows the bitrate estimation results comparing the estimated bitrate versus actual bitrate using another 18 30-seconds video clips encoded with different QPs. The correlation is 0.99 indicating the high accuracy of the proposed model.

Thus, Equations (4.10) $\sim$ (4.12) completes our bitrate model.

4.4 Optimization Algorithm

In the previous sections, we have proposed 1) a user experience model which models cloud mobile 3D display gaming user experience as a function of video encoding settings and graphics rendering settings, and 2) a bitrate model which estimates video bitrate needed to encode the rendered video as a function of video encoding settings and graphics rendering settings. In this section, we combine these two models so that by selecting proper graphics rendering (texture detail) settings and video encoding

(quantization parameter) settings of the rendered video, we can find an optimal solution for maximizing user experience (minimizing I) given a network bandwidth limit.

4.4.1 Problem Formulation

Figure 4.8 shows the problem formulation. We formulate the problem as an optimization problem. Because according to Equation (4.1) and (4.2), maximizing MOS is equal to minimizing I, we set our optimization target as minimizing I. $TD_{\min}$, $TD_{\max}$, $QP_{\min}$ and $QP_{\max}$ are the minimum and maximum boundaries of the settings being used for an application.

4.4.2 Algorithm Description

We first describe the key ideas and insights of how we analyze the problem and develop the algorithm. Then we discuss the detailed steps of the algorithm.

First, notice that the problem we are to solve contains 4 variables that are all discrete variables. For a convex or concave problem with continuous variables, it is very easy to solve. However, for the proposed problem, we are faced with two difficulties 1) It's hard to prove our problem is a convex or concave problem directly or maybe the problem is not convex or concave at all. 2) The variables are discrete. From our previous work, we can conclude that discrete variable linear programming is very hard to solve, discrete variable convex optimization is even harder and if the problem is not even convex or concave, it will be almost impossible to find a shortcut and prove optimality.

Notice that assuming the variable range space of TD is n, the combination of TD_L and TD_R can only have $2n-1$ choices. That is because we only allow either $TD_L = TD_R$ or $TD_L = TD_R - 1$ as in our previous work our subjective test shows when TD_L and TD_R has more than one level difference, it creates a large impairment, so we exclude that case from our value space. In addition, in reality, n can only be a very small number as

it is not possible to define too much texture detail levels. In our experiment, we choose $n = 3$ resulting in 5 combinations in total that is two orders less than the number of combinations of QP_L and QP_R. Thus, we propose to divide our problem into $2n-1$ sub-problems with only two variables, calculate the best solution for each sub-problem and compare to get the final best solution. In this case, one advantage is that the number of variables are decreased so that it becomes possible to prove the sub-problem to be a convex or concave problem. Note that the objective function is a piecewise function; we provide proof in Section 4.4.3 to show that one piece of the objective function together with the constraint functions is a convex problem. Thus, we propose to first relax the discrete variable optimization problem into a continuous variable optimization problem. Then we propose to use a relatively small search space that we will discuss in detail later to find the real optimal discrete solution. We can also prove the complexity of the search space is $O(m)$ where m is the possible choices of QP. By utilizing the thoughts and ideas above, we are able to design an algorithm leading to an optimal solution with low complexity.

In the following, we describe the details of the algorithm. Figure 4.9 shows the block diagram of our proposed algorithm. At the beginning, in step 1, we will first compute the values of I for all combinations of the parameters (QP_L, QP_R, TD_L, TD_R) and sort them in a queue called Q_S. As this is a one-time computation and the result can be saved in memory, it saves redundant computations during the execution of the algorithm, and at the same time the sorting itself will be an important step in this algorithm. The rest of the algorithm is designed to be periodically executed according to a fixed time interval. In our experiment, we use 1 second as the interval. During each interval, in step 2, video content features are extracted to estimate the video bitrate, and network bandwidth is estimated through a network probing method proposed in our previous work. Then, at the end of each time interval, after gathering the necessary

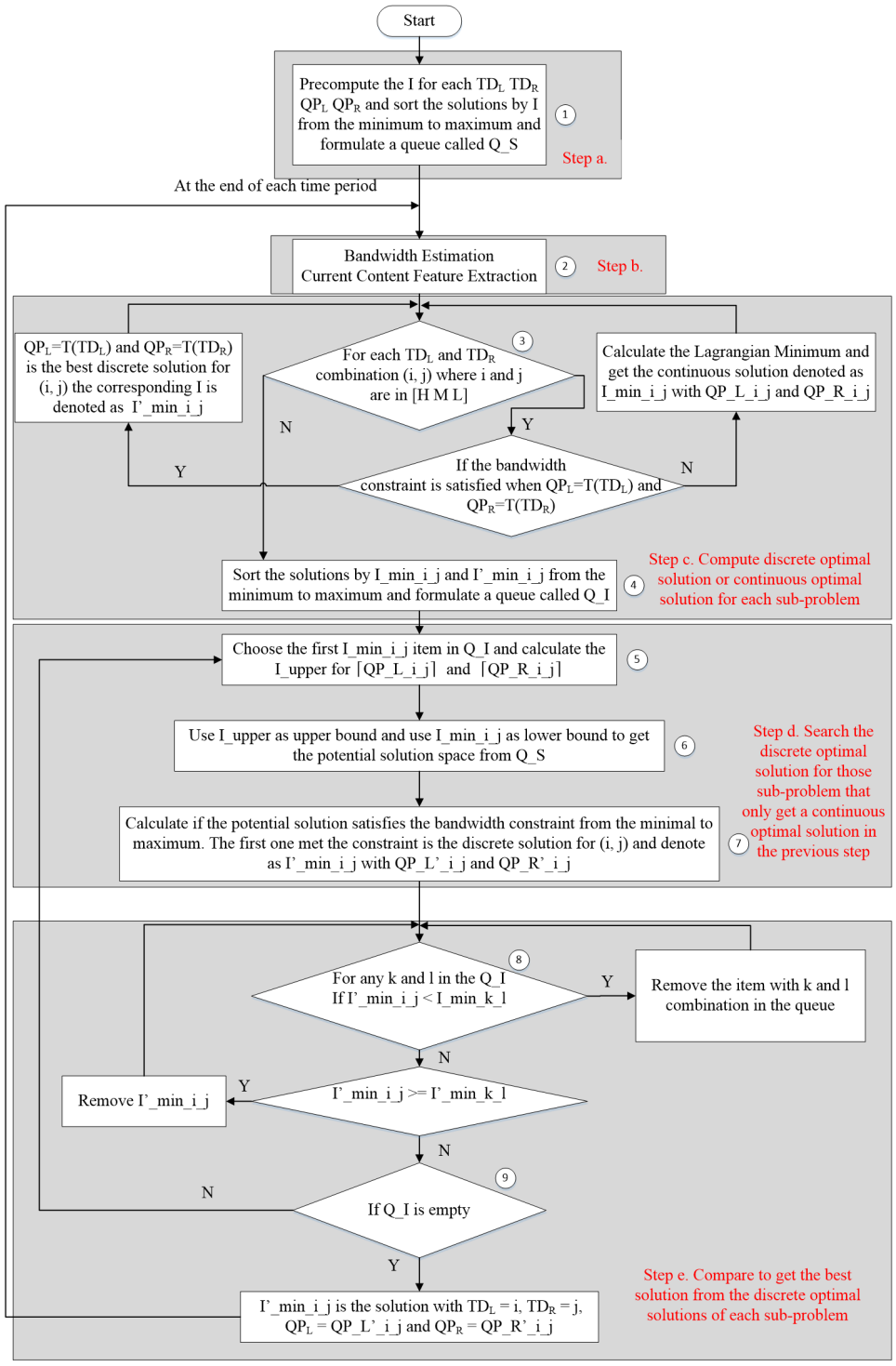


Figure 4.9. Validation of bitrate estimation

information, we divide our problem into $2n-1$ sub-problems. For each sub-problem, in step 3, we first check whether the condition when $QP_L = T(TD_L)$ and $QP_R = T(TD_R)$ satisfies the bandwidth constraint. For asymmetric settings of texture detail, for example $TD_L = M$ and $TD_R = L$, we will also check the conditions when $QP_L > T(TD_L)$ and $QP_R < T(TD_R)$. If these conditions satisfy the bandwidth constraint, the best one with lowest I among them will be the optimal discrete solution for the sub-problem because any other solution (described by g function in Equation (4.4)) in this sub-problem will result in higher I. Otherwise, we relax the discrete optimization problem into a corresponding continuous optimization problem and use Lagrangian Minimum method to get the solution for this continuous problem. The proof why the continuous problem is convex and the corresponding equations of how to compute the minimum are provided in Section 4.4.3. After getting the continuous minimum or discrete minimum for each sub-problem, we need to find out the real discrete minimum among all sub-problems. In step 4, we sort these solutions from minimal I to maximal I and put them in a queue called Q_I and then start examining from the minimal continuous solution. Notice that for any continuous solution whose corresponding parameters are $QP\_L\_i\_j$ and $QP\_R\_i\_j$ where L means left view, R means right view, i means the choice for TD_L and j means the choice for TD_R, the discrete solution when $QP\_L = QP\_L\_i\_j$ and $QP\_R = QP\_R\_i\_j$ will always satisfy the bandwidth constraint because both QPs are increasing, resulting in the bandwidth consumption to be decreasing. In step 5, we can set this solution as the upper bound and set the continuous solution as the lower bound for this sub-problem. By using this upper bound and lower bound, in step 6, we go back to Q_S and can select the solutions that have lower I than upper bound and higher I than lower bound. In step 7, we compute from the lowest I to the highest I to see if one of those solutions can satisfy the bandwidth constraint and if so the first one that can satisfy the constraint will be the best discrete solution for this sub-problem. Note that in the worst case, the upper bound

will be the discrete solution so it is guaranteed to find a solution by this method. It is also proved in Section 4.4.3 that the complexity of searching for the discrete solution inside the bounds is $O(m)$ where m is the number of choices of QP. After getting one discrete solution, in step 8, the algorithm will compare it with the continuous or discrete solutions of the other sub-problems. If the current discrete solution is better than the continuous or discrete solutions of the other sub-problems, these solutions in other sub-problems do not need to be further examined to find the corresponding discrete solution. If the current discrete solution is not better than the discrete solutions in other sub-problems, the current solution will be dropped. This pruning will not result in inaccuracy but it will improve the execution time of the algorithm significantly. After the above in step 9, it will judge if Q_I is empty, if so, it can get the final optimal discrete solution for the problem.

4.4.3 Proofs

In the following, we will first prove that for each sub-problem, when g function applies, the problem is a convex problem and when it relaxes into a continuous problem, it can be solved by Lagrangian Minimum method. We will also derive the equations of how to compute the minimal solution.

According to [77], an optimization problem of the form

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g_i(x) \leq 0, i = 1, \ldots, n \end{aligned} \tag{4.13}$$

is called convex if the functions $f, g_1, ..., g_m : R^n \to R$ are convex. [77]

Recall that the objective function when g function applies is

$$g(x_1, x_2) = a_1 x_1 + a_2 x_2 + b_1 \tag{4.14}$$

where $x_1 = QP_L - T(TD_L)$ and $x_2 = QP_R - T(TD_R)$

For a given sub-problem, TD_L and TD_R are fixed. Thus

$$g(QP_L, QP_R) = c_1 QP_L + c_2 QP_R + c_3 \tag{4.15}$$

Since the g function is a bilinear function, it can be considered as both a convex function and a concave function.

In addition, for a given sub-problem, TD_L and TD_R are fixed, the constraint function is

$$R(QP_L, QP_R) = R(QP_L) + R(QP_R) \tag{4.16}$$

If we combine all the constants together, the constraint function can be rewritten as

$$R(QP_L, QP_R) = e_1 2^{d_1(QP_L - 4)} + e_2 2^{d_2(QP_R - 4)} \tag{4.17}$$

Because the Hessian Matrix of the above equation is

$$H = \begin{bmatrix} (d_1 \ln 2)^2 e_1 2^{d_1(QP_L-4)} & 0 \\ 0 & (d_2 \ln 2)^2 e_2 2^{d_2(QP_R-4)} \end{bmatrix} \tag{4.18}$$

And also because $e_1 > 0$ and $e_2 > 0$, H is a positive definite matrix. Thus, the constraint function is a convex function.

The range of QPs are from 0 to 51 and we also have $QP_L \leq QP_R$. Thus, the range of the variable is a closed convex set.

Therefore, our problem is a convex optimization problem. After proving that the problem is a convex optimization problem, we can compute the minimal value by the following method.

First, construct an F function as

$$F\left(QP_L, QP_R\right) = g\left(QP_L, QP_R\right) + \lambda\left(R\left(QP_L, QP_R\right) - BW\right) \tag{4.19}$$

Let

$$\begin{cases} \frac{\partial g(QP_L, QP_R)}{\partial QP_L} + \lambda \frac{\partial R(QP_L, QP_R)}{\partial QP_L} = 0 \\ \frac{\partial g(QP_L, QP_R)}{\partial QP_R} + \lambda \frac{\partial R(QP_L, QP_R)}{\partial QP_R} = 0 \\ R\left(QP_L, QP_R\right) = BW \end{cases} \tag{4.20}$$

We can solve that

$$\begin{aligned} QP_L &= \frac{\log_2\left(\frac{BW}{e_1} \cdot \frac{c_1}{c_1+c_2}\right)}{d_1} + 4 \\ QP_R &= \frac{\log_2\left(\frac{BW}{e_1} \cdot \frac{c_2}{c_1+c_2}\right)}{d_1} + 4 \end{aligned} \tag{4.21}$$

Thus, we prove that the problem is convex and also provide the explicit equation to solve the convex problem and find the optimized solution.

In the following, we will prove the search space of the best discrete solution of a sub-problem is of the complexity m where m is the variable range of QP.

Define:

$$g\left(x, y\right) = ax + by + c$$

Assume:
1. Both $a > 0$ and $b > 0$
2. x and y are both integers and both ranges in $(0\ t]$.

Consider:
$\forall\left(x, y\right), \exists k$ pairs of (x', y')in total
s.t. $f(x,y) \leq f(x', y') \leq f(x+1, y+1)$

To prove:
$k \leq mt$, where m is a constant

Figure 4.10. Optimization problem

We first generalize the problem and prove the more general problem, so that our problem is just a special case under this framework. We can state the generalized problem below.

Then we provide proofs to this problem in the following:

$f(x,y) \le f(x',y') \le f(x+1,y+1)$

$\Leftrightarrow ax+by \le ax'+by' \le ax+by+a+b$

$\Leftrightarrow x+\frac{b}{a}y \le x'+\frac{b}{a}y' \le x+\frac{b}{a}y+1+\frac{b}{a}$

Let $\Delta x = x'-x$

$\Delta y = y'-y$

$\Leftrightarrow 0 \le \Delta x+\frac{b}{a}\Delta y \le 1+\frac{b}{a}$

$\Leftrightarrow -\frac{a}{b}\Delta x \le \Delta y \le 1+\frac{a}{b}-\frac{a}{b}\Delta x$

$\therefore$ For any Δx,

Δy can have at most $1+\frac{a}{b}$ values

$\because x' = x+\Delta x \in (0\ t]$

$y' = y+\Delta y \in (0\ t]$

$\therefore \Delta x \in [-x\ t-x]$

$\Delta y \in [-x\ t-x]$

$\because$ Both x and x' are integers

$\therefore \Delta x$ is also an integer

$\therefore \Delta x$ can have t values

$\therefore \Delta y$ can have at most $t\left(1+\frac{a}{b}\right)$ values

$\therefore \mathrm{k} \le t\left(1+\frac{a}{b}\right)$

where $1+\frac{a}{b}$ is a constant

4.4.4 Complexity Analysis

We define the range of the parameter TD as n and that of QP as m. Because each parameter can be set in both left view and right view, considering the brute force algorithm, the complexity of the original problem is $O(n^2m^2)$.

In the proposed algorithm, for each sub-problem, we calculate the continuous optimal and then search for the discrete optimal in a space of the complexity m. Thus, the worst case complexity of this algorithm is $O(n^2m)$. Considering the pruning we introduced in the algorithm that if the first discrete solution is better than all the other the continuous solutions, we do not need to search discrete solutions for other sub-problems, the complexity of the algorithm can be reduced to $O(n^2+m)$. In our simulation of randomly generated data, we see that the algorithm prunes successfully in more than 90% of the cases.

4.5 Experimental Results

In this section, we report on experiments conducted using a commercial cloud service, Amazon Web Service (AWS) [78], to verify the performance improvement by applying the proposed JAVRE technique. We use the same testbed as shown in Figure 4.2, except that 1) we put a network emulator Linktropy [79] between AP and laptop to control wireless network condition and 2) we implement our CMVIA(3D) system, including the JAVRE algorithm, on AWS servers. In detail, we modified the open-source game engine Planeshift and open-source virtual world application SecondLife so that 1) the rendering engine is able to pre-load different levels of textures when initializing the rendering loop and 2) when one specific texture detail setting is chosen it is able to switch to it dynamically. We also programmed a control software that is able to 1) capture the game scene or virtual world scene in real time, 2) encode the left and right view videos by x265

library, 3) stream the videos to the client devices through TCP, 4) probe the network and estimate the available network bandwidth and 5) with the JAVRE algorithm implemented inside, decide the parameters TDs and QPs that result in best use experience and set them in the rendering engine through process level share memory mechanism

For the Amazon cloud server, the CPU is Intel Xeon E5-2670 @2.60GHz with 15GB memory and the GPU has 1536 CUDA cores and 4GB of video memory. The operating system is Windows Server 2008 R2 SP1.

We firstly collected real 4G-LTE network traces by using network bandwidth testing software Speedtest.net [74] to record the bandwidth. Figure 4.11 shows a sample LTE trace, which is emulated using the network emulator in our testbed. We then measured the bandwidth from Amazon cloud server to our lab. We use iPerf software to test, and collect bandwidth value every 10 minutes from 8:00 a.m. to 10:00 p.m. for three days. Figure 4.12 shows the PDF of the results. We find that the bandwidth has some variance but is quite adequate. Compared to Figure 4.11 where the largest bandwidth of LTE is about 3.4Mbps, the lowest bandwidth from AWS to our lab is about 5.8Mbps. Thus we conclude the bandwidth bottleneck on the entire transmission flow is caused by the LTE trace. In addition, for comparison reasons, we also implemented two other algorithms called ARA from our previous work and JREA from [51]. Basically, ARA enables the game to set two different texture details for left view and right view, but video encoding settings are fixed to the highest values. ARA also enables view distance settings which let the game not render the objects whose distance to the virtual camera is greater than a certain threshold (view distance threshold). JREA is an adaptation algorithm developed for 2D CMG applications. The basic idea for this technique is that it pre-defines several groups of parameter combinations and assigns them into different levels. The algorithm chooses to go up a level or go down a level at a time when the network conditions changes. We extended the framework of it for CMVIA(3D), and

evaluated the performance of the algorithms on all three applications (gaming, virtual classroom and virtual art gallery).

Figures 4.13a $\sim$ 4.13c show the result for cloud mobile 3D display gaming while Figures 4.14a $\sim$ 4.14c show the result for cloud mobile 3D virtual classroom and Figures 4.15a $\sim$ 4.15c show the result for cloud mobile 3D virtual art gallery. Figures 4.13a,4.14a,4.15a plot the video bitrate while Figures 4.13b,4.14b,4.15b plot the corresponding impairment *I* and Figures 4.13c,4.14c,4.15c show the MOS. In each figure, we compare the performance of the three algorithms (JAVRE, ARA and JREA). The average values of I and MOS for each algorithm and each application are also shown in Table 4.4. From the figures and the table, we can make the following observations:

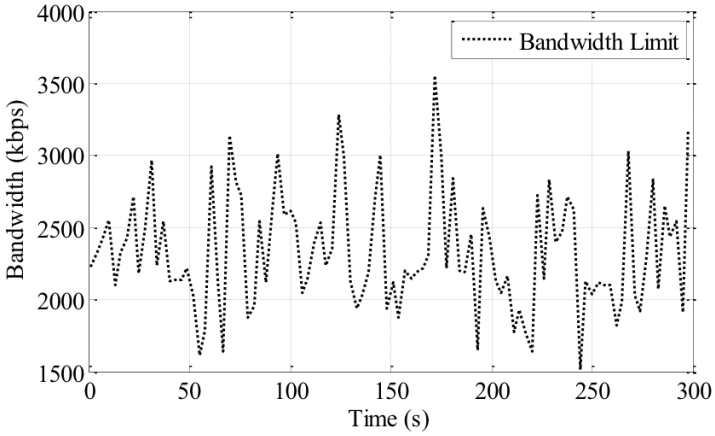


Figure 4.11. LTE bandwidth trace

Table 4.8. Statistical results of the experiment showing impairment *I* and overall user experience mos.

Application	*I*			MOS		
	JAVRE	ARA	JREA	JAVRE	ARA	JREA
Gaming	13.93	42.44	57.95	4.13	3.10	2.51
Virtual Classroom	7.64	26.59	40.31	4.35	3.71	3.16
Virtual Art Gallery	17.70	44.31	60.81	3.67	2.92	2.11

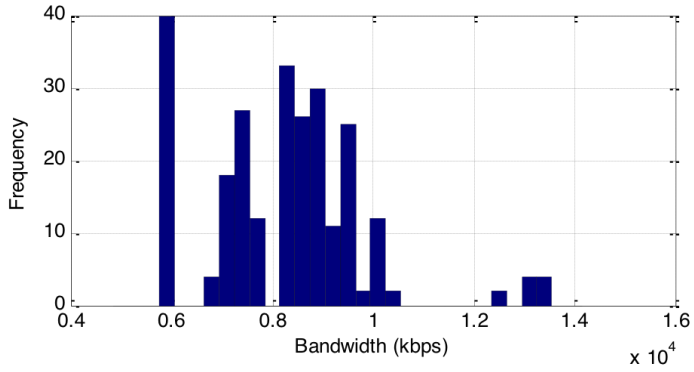
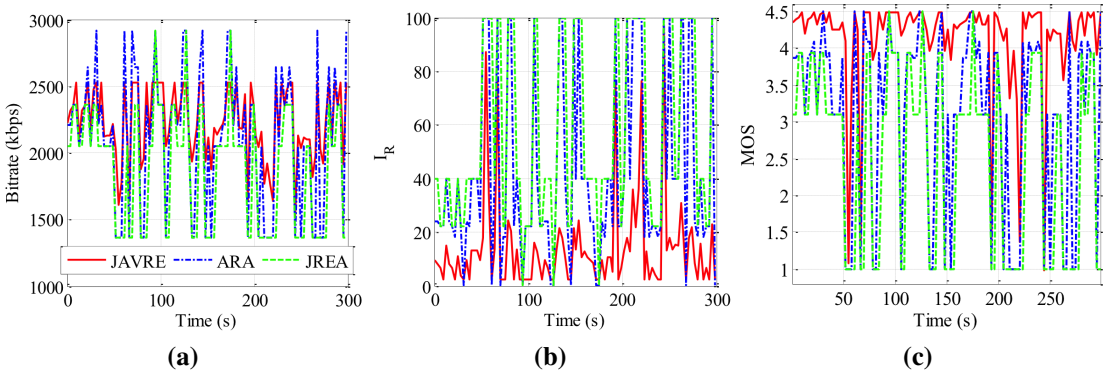


Figure 4.12. PDF of the bandwidth from Amazon cloud server to UCSD Mobile System Design Lab



(a) **(b)** **(c)**

Figure 4.13. (a) left, Bandwidth consumption of the algorithms for gaming (b) middle, Resulting impairment *I* of the algorithms for gaming (c) right, MOS of the algorithms for gaming

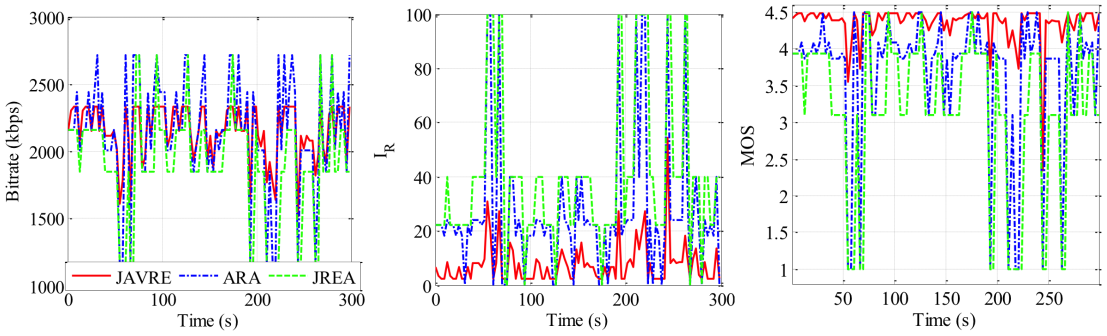


Figure 4.14. (a) left, Bandwidth consumption of the algorithms for virtual classroom (b) middle, Resulting impairment *I* of the algorithms for virtual classroom (c) right, MOS of the algorithms for virtual classroom

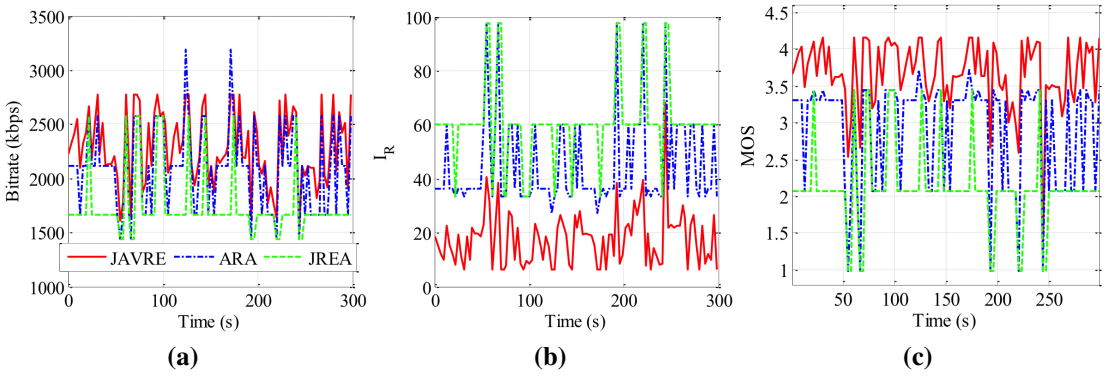


Figure 4.15. (a) left, Bandwidth consumption of the algorithms for virtual art gallery (b) middle, Resulting impairment *I* of the algorithms for virtual art gallery (c) right, MOS of the algorithms for virtual art gallery

1) In all applications, JAVRE performs the best (result in lowest I and highest MOS), ARA is the next and JREA is the worst. The improvement in terms of MOS by using JAVRE over ARA is up to 33.2% and that over JREA is up to 64.5%.

2) For all the three algorithms, the MOS values of gaming are all lower than that of virtual classroom. The reason is that a) the parameters of the user experience model for gaming and virtual classroom are very similar and b) for virtual classroom application, the virtual camera is mostly fixed with limited movement, unlike the gaming that the camera is mostly moving which make the video bitrate to be lower for the same parameter setting and therefore for the virtual classroom application, under the same bandwidth constraint, it can choose the TDs and QPs result in lower I and higher MOS.

3) For all the three algorithms, the MOS values of virtual art gallery are all lower than that of virtual classroom. The reason is in virtual art gallery application, the viewers pay extensive attention to the details of the virtual paintings and hence will be more sensitive to the influence of decreasing TD or increasing QP. Therefore, although for virtual art gallery application, the algorithm will choose the parameters resulting in more bandwidth consumption, but it still doesn't provide the user experience as well as it does for the virtual classroom application.

4.6 Conclusion

The main contributions of this chapter are the following.

1) We performed extensive subjective tests to derive a general user experience model for cloud mobile 3D virtual immersive applications considering both asymmetric video encoding and graphics rendering.

2) We derived a bitrate model which is suitable for H.265/HEVC standard, by taking into account both video encoding parameters and graphics rendering parameters.

3) We developed a novel adaptation algorithm called JAVRE that can decide

QPs and TDs dynamically by making use of the above two models to ensure best user experience for cloud mobile 3D virtual immersive applications under dynamic wireless network condition.

4) By conducted experiments using real 4G-LTE network profiles on commercial cloud service with 3 different virtual immersive applications, we demonstrated significant improvement over existing methods in user experience when the proposed JAVRE algorithm is applied.

4.7 Acknowledgements

Chapter 4, in part, is from the material as it appears in proceedings of IEEE ISM 2015. Yao Lu; Yao Liu; Sujit Dey. and in IEEE Journal on Emerging and Selected Topics in Circuits and Systems 2016. Yao Lu; Sujit Dey. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Conclusion and future work

5.1 Conclusion

In this thesis, we develop techniques to improve user experience for cloud mobile 3D applications. In each chapter, we start from conducting subjective experiment to derive a user experience model. A bit rate model is then proposed and the algorithm is designed to solve the optimization problem. The technologies proposed are evolving. We start by explore the possibility of enabling asymmetric graphics rendering for the entire frame (ARA). Then, we further experiment with the idea to enable asymmetric graphics rendering for each individual objects (ASORA). Further, jointly optimization by using asymmetric graphics rendering and asymmetric video encoding together(JREA) is investigated. The experimental results show that by using ARA, it is much better than traditional symmetric settings in terms of user experience. Also, by using ASORA and JREA, the user experience is further improved under fluctuating network conditions. Moreover, in the third work, we also extend the application from game only to other virtual immersive applications such as virtual classroom and virtual art gallery. To conclude, the techniques proposed in this thesis improve the overall user experience for cloud mobile 3D virtual immersive applications under challenging network conditions.

5.2 Future work

In the future, we would like to extend our research reported in this thesis in the following several directions.

First, more advanced video encoding tools need to be investigated. In our prototype we firstly use x264 that is an open-source implementation of H.264/AVC standard. We then moved on to use x265 [15] that is another open-source implementation of H.265/HEVC standard. These two encoders are all single view video encoder that do not take into account the redundant information between left view and right view. In order to make use of the this information, 3D-HEVC [16] and MV-HEVC [17] are two standards that need to be integrated and experimented with. For these two new standards, currently no real-time open-source implementation is available. Consequently, we will have to wait till such an implementation becomes available, or we will need to develop a software prototype of the new real-time encoder before applying it in the cloud gaming pipeline. Besides, traditional encoding standards are developed for nature scene videos captured by cameras. Computer graphics videos, on the other hand, are different. They are generated by computer algorithms running on GPUs. HEVC has recently announced its screen content encoding standard called HEVC-SCC [38]. It would be promising to see HEVC-SCC can also be integrated into the pipeline and achieve better performance. It can be foreseen that with these new encoding tools integrated, the compression efficiency will be increased and thus the user experience will be further improved.

Secondly, in Chapter 3 and Chapter 4, in our problem formulation, we assume our technique will ensure that the resulting video bitrate will not exceed the available bandwidth and hence there will be no network congestion and therefore no impairment due to wireless network delay. This is based on the network bandwidth estimation technique proposed by us in Chapter 2. This technique is proved to be effective for wired

networks which have very low possibility of packet loss in layer two. However, this may not be always achievable under wireless network conditions due to random packet loss. Hence, in the future, we will consider random packet loss in wireless networks and propose a better network bandwidth estimation scheme. We will also conduct more subjective tests to include network delay and network packet loss impairment in our user experience model.

Thirdly, our research can be extended to support virtual reality and augmented reality applications with Head Mounted Displays (HMD) such as Oculus Rift [9], HTC Vive [10], PlayStation VR [11], Samsung Gear VR [18], Microsoft Hololens [19] etc. Considering the fact that the screens of HMDs are very close to human eyes, visual impairment is more noticeable. Thus, the overall quality (graphics rendering quality plus video encoding quality) of the streamed content needs to be improved to ensure acceptable user experience. Besides, the asymmetric quality between two views may also introduce more impairment compared to viewing the content using a 3D monitor or a 3D mobile device. Another challenge for HMD is that it is highly delay sensitive. Every movement of the head will generate an input command to control the camera on the cloud that makes the requirement of the delay to be less than 20-30ms [20]. Otherwise, it can easily cause dizzy effect for human beings. One possible solution to overcome this problem is to put multiple virtual cameras in the scene, generate multiple views from different angles and stitch the video into a 360 degree video. In this way, it eliminates the requirement to send the head motion back to the server and therefore make the requirement of the delay to be less strict. GPU utilization of larger number of virtual cameras needs to be investigated and a real-time stitching algorithm needs to be developed. Further subjective tests also need to be performed to carefully model the user experience for virtual reality applications on HMDs and an adaptation algorithm needs to be developed for these cases.

Last but not least, this thesis focuses on how to adaptively tune various parameters to ensure best user experience, but it does not consider the cloud cost. The cloud cost contains two parts. One is bandwidth cost, and the other is hardware cost that is decided by the computational complexity of the system. In fact, almost all the parameters considered in this thesis are not only related to user experience but also related to the computational complexity. For example, if we set texture detail from Medium to High, it improves the graphics rendering quality, increases video bitrate but it will also increase the complexity or more specifically speaking, GPU utilization. Thus, larger the complexity, more the number of GPUs needed, and thus higher the cloud cost. To tackle this practical problem and make some tradeoffs between cloud cost and user experience, a computational complexity model needs to be developed to take into account all the parameters and compute how many GPU machines are needed for the whole task. In addition, considering that each application requires different resources but each GPU machine have fixed resource, a scheduling algorithm may also need to be designed to assign each application to a specific machine or machines to optimize the hardware usage. Some preliminary research on this topic can be investigated and referred to, such as [21] and [22].

To conclude, cloud architectures optimized for gaming and other graphics-heavy applications are recently emerging. To ensure good user experience is hard to achieve. To ensure good user experience while preserving low cost is an even more challenging task. This thesis focuses on how to ensure good user experience without considering cost. A lot of future work needs to be done both in terms of improving user experience and lowering cost so as to make this technology better and become a commercial success.

Bibliography

[1] Ryan Shea, Jiangchuan Liu, Edith C-H. Ngai, and Yong Cui. Cloud gaming: architecture and performance. IEEE Network 27(4):16-21, 2013

[2] Google Doc, https://apps.google.com/

[3] Amazon Web Service, https://aws.amazon.com/

[4] Dropbox, https://www.dropbox.com/

[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al. A view of cloud computing. ACM Communications 53(4):50-58, 2010

[6] Jacob Chakareski, Adaptive multiview video streaming: challenges and opportunities. IEEE Communications Magazine 51(5):94-100, 2013

[7] Grigore C. Burdea, and Philippe Coiffet. Virtual reality technology. Vol. 1. John Wiley & Sons, 2003

[8] Ronald T Azuma, A survey of augmented reality. Presence: Teleoperators and virtual environments 6(4):355-385, 1997

[9] Oculus, https://www.oculus.com/

[10] HTC Vive, https://www.htcvive.com/

[11] Sony Playstation VR, https://www.playstation.com/en-ca/explore/playstation-vr/

[12] Pocemon Go, https://www.pokemongo.com/

[13] Erik Dahlman, Stefan Parkvall, and Johan Skold. 4G: LTE/LTE-advanced for mobile broadband. Academic press, 2013.

[14] Yung?Yuan Kao, Yan?Pean Huang, Kai?Xian Yang, Paul C?P. Chao, Chi?Chung Tsai, and Chi?Neng Mo. 11.1: An Auto?Stereoscopic 3D Display Using Tunable Liquid Crystal Lens Array That Mimics Effects of GRIN Lenticular Lens Array. In SID symposium digest of technical papers, 40(1):111-114. Blackwell Publishing Ltd, 2009.

[15] x265, http://x265.org/

[16] Gerhard Tech, Ying Chen, Karsten Mller, Jens-Rainer Ohm, Anthony Vetro, and Ye-Kui Wang. Overview of the multiview and 3D extensions of High Efficiency Video Coding. IEEE Transactions on Circuits and Systems for Video Technology 26(1):35-49, 2016

[17] Gary J. Sullivan, Jill M. Boyce, Ying Chen, Jens-Rainer Ohm, C. Andrew Segall, and Anthony Vetro. Standardized extensions of high efficiency video coding (HEVC). IEEE Journal of selected topics in Signal Processing 7(6):1001-1016, 2013

[18] Samsung Gear VR, https://www.samsung.com/global/galaxy/gear-vr/

[19] Microsoft Hololens, https://www.microsoft.com/microsoft-hololens/en-us

[20] Jason D. Moss, Jon Austin, James Salley, Julie Coats, Krysten Williams, and Eric R. Muth. The effects of display delay on simulator sickness. Displays 32(4):159-168, 2011

[21] Zhengwei Qi, Jianguo Yao, Chao Zhang, Miao Yu, Zhizhou Yang, and Haibing Guan. VGRIS: virtualized GPU resource isolation and scheduling in cloud gaming. ACM Transactions on Architecture and Code Optimization (TACO) 11(2):17, 2014

[22] Chao Zhang, Jianguo Yao, Zhengwei Qi, Miao Yu, and Haibing Guan. vgasa: Adaptive scheduling algorithm of virtualized gpu resource in cloud gaming. IEEE Transactions on Parallel and Distributed Systems 25(11):3036-3045, 2014

[23] Maggie Shiels. “Console killer” OnLive to launch in June. http://news.bbc.co.uk/2/hi/technology/8556874.stm, 2010.

[24] Audrey Oeillet. Reportage : SFR dvoile son service de jeux vido “cloud gaming” sur Neufbox. http://www.clubic.com/connexion-internet/fai-sfr-box-neufbox/actualite-373750-cloud-gaming-sfr-service-jeux-video-neufbox.html, 2010.

[25] Rich Brown. Gaikai cloud-gaming service goes live. http://www.cnet.com/news/gaikai-cloud-gaming-service-goes-live/, 2011.

[26] Sharif Sakr. Sony buys Gaikai cloud gaming service for $380 million. https://www.engadget.com/2012/07/02/sony-buys-gaikai/, 2012.

[27] Chun-Ying Huang, De-Yu Chen, Cheng-Hsin Hsu, and Kuan-Ta Chen. Gaminganywhere: an open-source cloud gaming testbed. In Proceedings of the 21st ACM international conference on Multimedia, pp. 827-830. ACM, 2013.

[28] Eric Johnson. Sony acquires OnLives cloud gaming patents. http://www.recode.net/2015/4/2/11561104/sony-acquires-onlives-cloud-gaming-patents, 2015.

[29] Marshall Honorof. GeForce Now review: game streaming done (mostly) Right. http://www.tomsguide.com/us/geforce-now-game-streaming,review-3113. html/, 2015.

[30] Video codec for audiovisual services at px64 Kbit/s. ITU-T Rec H.261, version 1: Nov. 1990, version 2: Mar. 1993.

[31] Video coding for low-bit-rate communication. ITU-T Rec H.263, 1995.

[32] Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s Part 2: Video. ISO/IEC 11172-2 (MPEG-1), 1993.

[33] Coding of audio-visual objects Part 2: Visual. ISO/IEC 14496-2 (MPEG-4 Visual version 1), 1999.

[34] Generic coding of moving pictures and associated audio information Part 2: Video. ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG 2 Video), 1994.

[35] Advanced video coding for generic audio-visual services. ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), 2003.

[36] High efficiency video coding. ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC), 2013.

[37] Jizheng Xu, Rajan Joshi, and Robert A Cohen. Overview of the emerging HEVC screen content coding extension. IEEE Transactions on Circuits and Systems for Video Technology, 26(1):50-62, 2016.

[38] Mathias Wien, Renaud Cazoulat, Andreas Graffunder, Andreas Hutter, and Peter Amon. Real-time system for adaptive video streaming based on SVC. IEEE Transactions on Circuits and Systems for Video Technology, 17(9):1227-1237, 2007.

[39] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In Proceedings of the second annual ACM conference on Multimedia systems, pp. 157-168. ACM, 2011.

[40] Hamideh Afsarmanesh and Luis M Camarinha-Matos. Future smart-organizations: a virtual tourism enterprise. In Proceedings of IEEE International Conference on Web Information Systems Engineering, volume 1, pp. 456-461, 2000.

[41] Gorkem Saygili, Cihat Goktug Gurler, and A Murat Tekalp. Evaluation of asymmetric stereo video coding and rate scaling for adaptive 3D video streaming. IEEE Transactions on Broadcasting, 57(2):593-601, 2011.

[42] Sima Valizadeh, Maryam Azimi, and Panos Nasiopoulos. Bitrate reduction in asymmetric stereoscopic video with low-pass filtered slices. In Proceedings of IEEE International Conference on Consumer Electronics (ICCE), pp. 170-171, 2012.

[43] Bert Vankeirsbilck,Tim Verbelen,Dieter Verslype,Nicolas Staelens,Filip De Turck, Piet Demeester, and Bart Dhoedt. Quality of experience driven control of interactive media stream parameters. In Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, pp. 1282-1287, 2013.

[44] Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hossfeld. An evaluation of QoE in cloud gaming based on subjective tests. In Proceedings of IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 330-335, 2011.

[45] Kuan-Ta Chen, Yu-Chun Chang, Hwai-Jung Hsu, De-Yu Chen, Chun-Ying Huang, and Cheng-Hsin Hsu. On the quality of service of cloud gaming systems. IEEE Transactions on Multimedia, 16(2):480-495, 2014.

[46] Yao Liu, Shaoxuan Wang, and Sujit 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, 4(1):43-56, 2014.

[47] Chengsheng Que, Guobin Chen, and Jilin Liu. An efficient two-pass VBR encoding algorithm for H.264. In Proceedings of IEEE International Conference on Communications, Circuits and Systems, pp. 118-122, 2006.

[48] Zhan Ma, Meng Xu, Yen-Fu Ou, and Yao Wang. Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization stepsize and its applications. IEEE Transactions on Circuits and Systems for Video Technology, 22(5):671-682, 2012.

[49] Bumshik Lee and Munchurl Kim. Modeling rates and distortions based on a mixture of Laplacian distributions for inter-predicted residues in quadtree coding of HEVC. IEEE signal processing letters 18(10):571-574, 2011

[50] Mahdi Hemmati, Abbas Javadtalab, Ali Asghar Nazari Shirehjini, Shervin Shirmohammadi, and Tarik Arici. Game as video: Bit rate reduction through adaptive object encoding. In Proceeding of ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 712, 2013.

[51] Shaoxuan Wang and Sujit Dey. Adaptive mobile cloud computing to enable rich mobile multimedia applications. IEEE Transactions on Multimedia, 15(4):870-883, 2013.

[52] Shaoxuan Wang and Sujit Dey. Cloud mobile gaming: modeling and measuring user experience in mobile wireless networks. In Proceedings of ACM SIGMOBILE Mobile Computing and Communications Review 16(1):10-21, 2012

[53] Shaoxuan Wang and Sujit Dey. Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming, In Proceedings of IEEE Global Communications Conference (GLOBECOM), Miami, 2010.

[54] Yao Liu, Shaoxuan Wang, and Sujit Dey. Modeling, Characterizing, and Enhancing User Experience in Cloud Mobile Rendering. In Proceedings of IEEE International Conference on Computing, Networking and Communications (ICNC), Maui, 2012.

[55] Vinay Joseph Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In Proceedings of Passive and Active Measurement Workshop, 2003.

[56] Svante Ekelin, Martin Nilsson, Erik Hartikainen, Andreas Johnsson, J-E. Mangs, Bob Melander, and Mats Bjorkman. Real-time measurement of end-to-end available bandwidth using kalman filtering. In Proceedings of IEEE/IFIP Network Operations and Management Symposium(NOMS), pp. 73-84, 2006.

[57] PlaneShift, http://www.planeshift.it/

[58] ITU-T Rec. G.107, The E-model, a computational model for use in transmission planning, Mar, 2005.

[59] Sujit Dey, Yao Liu, Shaoxuan Wang, and Yao Lu, Addressing response time of cloud-based mobile applications. In Proceedings of ACM International Workshop on Mobile Cloud Computing & Networking, pp. 3-10, 2013

[60] Nang, Jongho, Sangchul Kim, and Hyuk-Jun Lee. Classifying useful motion vectors for efficient frame rate up conversion of MC-DCT encoded video streams. Journal of Information Science and Engineering 30(6)1755-1771, 2014

[61] R. Schreier and A. Rothermel. Motion adaptive intra refresh for the H. 264 video coding standard, IEEE Transactions on Consumer Electronics, 52(1):249-253, 2006

[62] M. Bremicker, Panos Y. Papalambros, and H. T. Loh. Solution of mixed-discrete structural optimization problems with a new sequential linearization algorithm. Computers & Structures, 37(4):451-461, 1990

[63] Broadsides, http://cse125.ucsd.edu/cse125/2012/cse125g1/

[64] A. E. Eckberg, Approximations for bursty (and smoothed) arrival queueing delays based on generalized peakedness, In Proceedings of International Teletraffic Congress, Kyoto, 1985.

[65] Kerry W. Fendick and Ward Whitt. Measurements and approximations to describe the offered traffic and predict the average workload in a single-server queue. In Proceedings of the IEEE, 77(1):171-194, 1989

[66] BT-500-11: Methodology for subjective assessment of the quality of television picture, International Telecommunication Union.

[67] Yao Lu, Yao Liu and Sujit Dey, Optimizing Cloud Mobile 3D Display Gaming User Experience by Asymmetric Object of Interest Rendering. In Proceedings of IEEE International Conference on Communication (ICC), 2015

[68] Yang Liu, Zheng Guo Li, and Yeng Chai Soh. Region-of-interest based resource allocation for conversational video communication of H. 264/AVC. IEEE transactions on circuits and systems for video technology, 18(1):134-139, 2008

[69] Dimitris Agrafiotis, David R. Bull, Nishan Canagarajah and Nawat Kamnoonwatana. Multiple priority region of interest coding with H. 264. In Proceedings of International Conference on Image Processing (ICIP), pp. 53-56, 2006.

[70] Yen-Fu Ou, Zhan Ma, Tao Liu, and Yao Wang. Perceptual quality assessment of video considering both frame rate and quantization artifacts. IEEE Transactions on Circuits and Systems for Video Technology, 21(3):286-298, 2011

[71] x264, http://www.videolan.org/developers/x264.html/, Jun. 2015

[72] Prabhakant Sinha, and Andris A. Zoltners. The multiple-choice knapsack problem. Operations Research, 27(3):503-515, 1979

[73] Paul C. Chu, and John E. Beasley. A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics, 4(1):63-86, 1998

[74] Speedtest.net, https://itunes.apple.com/us/app/speedtest.net-mobile-speed/id300704847?mt=8/, Jun. 2015

[75] IBM ILOG CPLEX optimizer, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, Jun. 2015

[76] SecondLife, http:// www.secondlife.com/

[77] S. Boyd, and L. Vandenberghe, Convex optimization, Cambridge university press, 2004

[78] Amazon Web Service, http://aws.amazon.com

[79] Linktropy, http://www.apposite-tech.com/products/