Cloud Mobile 3D Display Gaming User Experience Modeling and Optimization by Asymmetric Graphics Rendering

Yao Lu, *Student Member, IEEE*, Yao Liu, *Member, IEEE*, and Sujit Dey, *Fellow, IEEE*

***Abstract*—With the arrival of auto-stereoscopic 3D displays for mobile devices, and emergence of more 3D content, there is much anticipation for 3D mobile multimedia experiences, including 3D display gaming. Simultaneously, with the emergence of cloud computing, more mobile applications are being developed to take advantage of the elastic cloud resources. In this paper, we explore the possibility of developing Cloud Mobile 3D Display Gaming, where the 3D video rendering and encoding is performed on cloud servers, with the resulting 3D video streamed over wireless networks to mobile devices with 3D displays for a true 3D mobile gaming experience. However, with the significantly higher bitrate requirement for 3D video, ensuring user experience may be a challenge, both in terms of 3D video quality and network delay (response time), considering the bandwidth constraints and fluctuations of wireless networks. In this paper, we propose a new asymmetric graphics rendering approach which can significantly reduce the video encoding bitrate needed for a certain video quality, thereby making it easier to transmit the video over wireless network. However, since asymmetric graphics rendering may also impair the graphics quality, we need to be able to understand and measure its impact. We conduct subjective tests to study and model the impairments due to asymmetric graphics rendering and network delay, thereby developing a user experience model for cloud based mobile 3D display gaming. By conducting subsequent subjective tests, we prove the correctness of the impairment functions and the resulting user experience model. Furthermore, given any network condition, we propose to solve the problem of selecting the optimal graphics rendering factors for the left and right views so as to maximize user experience of cloud mobile 3D display gaming. In order to solve this problem, we first develop a model to estimate the resulting video bitrate of the rendered 3D video when certain graphics rendering factors are used. Next, we derive a model to predict the delay given the available network bandwidth and the video bitrate of the rendered 3D video. We use the above two models together with a branch and bound algorithm to solve the optimization problem and determine the optimal values for the left and right view rendering factors. Experiments conducted using real 4G-LTE network profiles on commercial cloud service demonstrate the feasibility of significant improvement in user experience when the proposed optimization algorithm is used to dynamically select optimal rendering factors according to changing network conditions.**

***Index Terms*—3D, asymmetric graphics rendering, branch and bound, cloud Mobile Gaming, subjective testing, user experience.**

Manuscript received April 23, 2014; revised August 28, 2014; accepted January 06, 2015. Date of publication January 23, 2015; date of current version March 18, 2015. This work was supported by the National Science Foundation under Grant CCF-1160832. The guest editor coordinating the review of this manuscript and approving it for publication was Prof. Patrick Le Callet.

The authors are with the Mobile Systems Design Lab, Electrical and Computer Engineering Department, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: luyao@ucsd.edu; yal019@ucsd.edu; dey@ece.ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2015.2396475

I. Introduction

In recent years, emergence of rich mobile devices and broadband mobile networks have led to the widespread development and use of mobile applications. 3D games have become one of the most popular mobile applications; recently being in the top-5 rankings in terms of number of downloads in both iOS App Store and Android Google Play Store [1]. While the user experience of current 3D games is limited by 2D screens, there has been a gradual trend by mobile device manufacturers to introduce 3D screens, for example several new 3D tablets like Commander 3D and NEO3DO released in 2013 [2], raising the prospects for richer and more immersive stereoscopic multimedia experiences including enabling true 3D gaming experiences.

A key challenge to enabling rich 3D gaming on mobile devices is the high computing requirements, leading to the current development and use of less rich, mobile specific games. It has been shown that even though the CPU and GPU capabilities of mobile devices keep getting upgraded, there is a growing gap with the requirements of rich Internet/console 3D games that are being developed [4] using the latest graphics rendering innovations. Moreover, battery life can be a big challenge, especially when playing graphics intensive games. Evolution of 3D games displayed on 3D screens can only exacerbate the computing and battery life challenges.

To address the problems mentioned above, Cloud Mobile Gaming (CMG) has been recently proposed as an alternative approach to enable rich Internet gaming on mobile devices [16]–[18]. Authors of [34] did some initial study on latency, energy and cost to prove the feasibility of this architecture. In CMG, the most compute intensive tasks of 3D rendering are performed on cloud servers, with the rendered video encoded and streamed from a cloud server to the mobile device, in response to gaming commands from the mobile device. Consequently, the CMG approach provides cross-platform, thin client, battery life extended solution for mobile gaming. However, because of the large amount of video data to be streamed, using the CMG approach shifts the challenge from computing and battery constraints of mobile devices to the fluctuating bandwidth constraint of the network. Thus, previous work in Cloud Mobile Gaming has focused on how to model user experience and how to optimize user experience given certain network conditions [3]–[5], [31]. However, the above techniques are based on rendering 2D videos and streaming

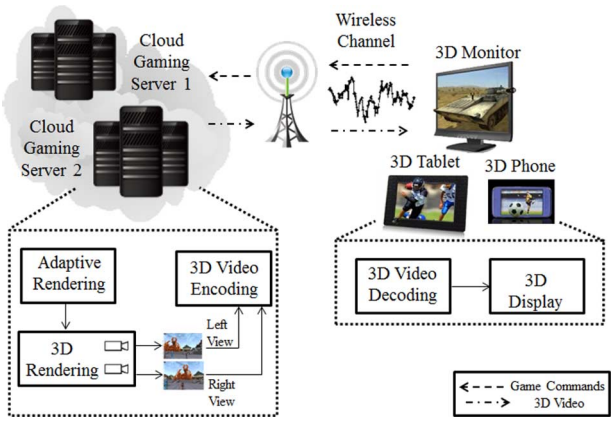

Fig. 1. Architecture of Cloud based Mobile 3D Display Gaming System.

to mobile devices with 2D screens. In this paper, we propose Cloud Mobile 3D Display Gaming (CMG(3D)), which renders 3D video with two virtual cameras, which is encoded and streamed to a mobile device with a 3D screen for a 3D viewing experience.

Fig. 1 shows the overall architecture and data flow of the proposed CMG (3D) platform. The 3D rendering is performed on cloud servers in response to gaming commands from the mobile device. We place two virtual cameras in the game world to generate a left view and a right view of game video. After the two views are generated, they will be encoded as a 3D video and transmitted through wireless network to the mobile device, and displayed on the device 3D screen. On the reverse side, game commands are transmitted from mobile device to the game server through wireless network. In this way, users can play 3D games as if the game is rendered locally, but with the advantages of a thin client, no need to download each game to the mobile device, and the game developers can develop a single version of the game which runs on the cloud servers instead of having to develop platform/device specific versions.

However, constrained and fluctuating network conditions can create significant challenge for the CMG(3D) architecture described above. Note that for gaming applications, especially those which are highly response time sensitive, transmitting the target video with low latency becomes the key factor which influences user experience [8]. We did an initial study of how large the delay will be under real 4G-LTE mobile network conditions. Fig. 2(a) shows a snapshot of a network profile which we captured using Iperf client software on iPhone and Iperf server software on remote CMG server during peak hours at UCSD. It describes how bandwidth is fluctuating during a time period of 300 seconds. We play an online open-source MMORPG game PlaneShift [9] using the CMG(3D) platform we have developed (Fig. 1), with the Cloud rendered 3D video streamed to the end device through a network emulator [27] programmed to emulate the bandwidth trace shown in Fig. 2(a). With the above test set-up, we measured response time which includes upload and download time shown in Fig. 2(b). Also shown is the video quality, measured as PSNR. For this kind of MMORPG game, the acceptable response time is 440ms and excellent delay is 120ms from a user study reported in [10]. As can be seen from

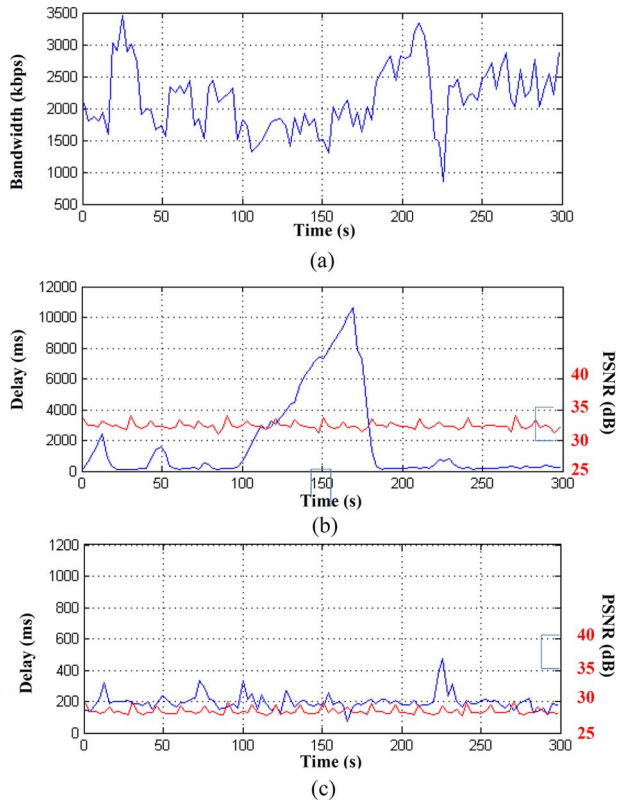

Fig. 2. (a) top, Network profile (b) middle, Delay and PSNR performance when using high quality video encoding (c) bottom, Delay and PSNR performance when using low quality video encoding.

Fig. 2(b), the experienced delay can be much higher than acceptable threshold. To solve this delay problem, we can change the video encoding setting and tradeoff video quality and delay. We show in Fig. 2(c) that by decreasing the video quality by an average of 4 dB, the experienced delay can be reduced. But the low video quality also impacts user experience greatly. From this example, we can see that it can be challenging to maintain good 3D video quality and response time simultaneously, under constrained and fluctuating wireless network conditions [19], [20].

Asymmetric 3D video encoding [12], [13], where the videos of the left and right views are encoded with different bitrates, can be potentially used to reduce the 3D video bit rate, and hence the delay, while transmitting over constrained wireless networks. From previous subjective studies [12], [13], we conclude that we can potentially use asymmetric video encoding to reduce 3D video bit rate in the range of 10%-30% without compromising overall video quality. However, the above may not be enough to stream high bit rate high motion 3D rendered gaming video over fluctuating wireless networks. Hence, in this paper we investigate an alternative and complimentary approach, which as shown in Fig. 4 has the potential to reduce bitrate much more significantly without compromising video quality.

Previous research with cloud mobile gaming for 2D mobile displays (where video streamed from cloud to mobile device is 2D as opposed to 3D video considered in this paper for 3D displays) has been done to measure and optimize user experience by adapting video encoding parameters [32], [33]. In addition, another study has shown that adapting the graphics rendering

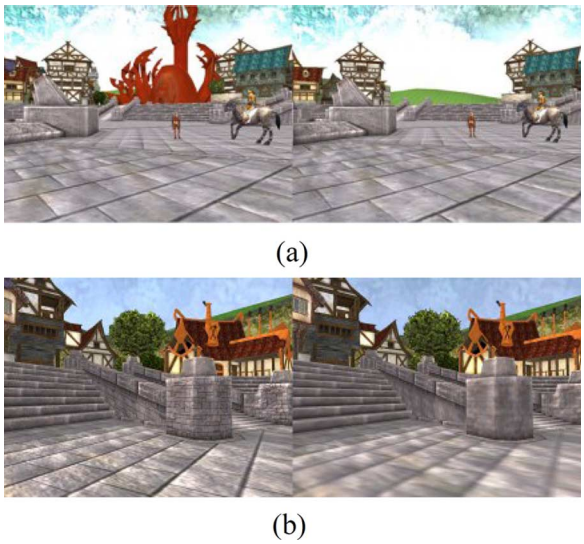

Fig. 3. (a) top, left view and right view with asymmetric view distance (b) bottom, left view and right view with asymmetric texture detail.

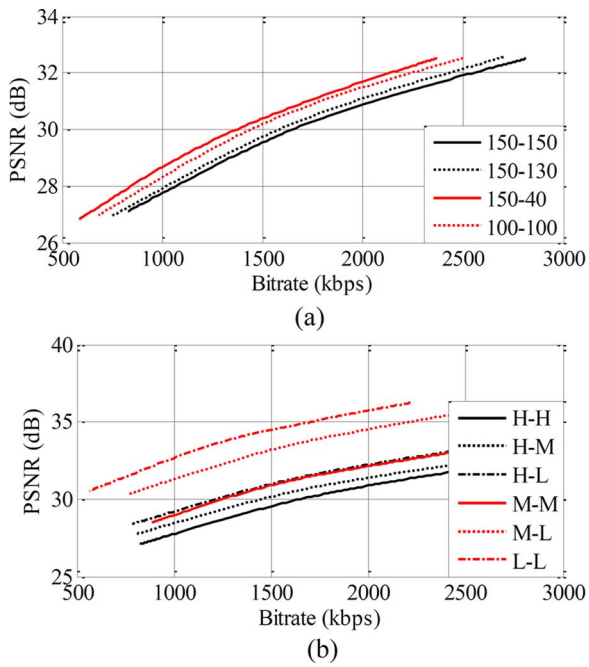

Fig. 4. (a) top, Rate-distortion performance with different view distance setting for left view and right view (b) bottom, Rate-distortion performance with different texture detail settings for left view and right view.

richness can significantly reduce the encoding bitrate needed for the resulting game video, and hence can enable streaming of high quality 2D video over constrained wireless networks while meeting delay constraints [4], [5], [28]. In this paper, to address the challenge of streaming high quality 3D gaming video with low delay constraint, we propose a new *asymmetric graphics rendering* approach where the rendering richness of one of the views can be degraded below that of the primary view. We demonstrate that our proposed approach can significantly reduce the 3D video encoding bitrate needed for a desired video quality, thereby making it possible to transmit the 3D video over wireless networks with the desired user experience of high video quality while meeting low delay constraint.

However asymmetric graphics rendering technique may cause graphics artifacts, which may also influence user experience. Hence we develop a model for Cloud mobile 3D display gaming user experience (CMG(3D)-UE) to quantitatively measure user experience in the presence of asymmetric graphics rendering. We further consider how to apply this model and automatically select the optimal graphics rendering factors so that the user experience can be maximized. We formulate the optimization problem as a discrete variable, non-linear objective function, with non-linear constraints. To solve the problem, we develop a model to estimate the bitrate of the resulting encoded 3D video stream given graphics rendering factors selected for left and right views, and another model to predict the delay given the available network bandwidth and the video bitrate of the rendered 3D video. We develop a branch and bound based algorithm which uses the above models to solve the optimization problem and provide the optimal values for the left and right view rendering factors. The above algorithm will be executed periodically and is applied at the beginning of each time interval during a CMG(3D) gaming session (which is the duration between when a player starts playing the game and ends playing the game), providing 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) gaming session. The rest of the paper is organized as following: Section II introduces details about asymmetric graphics rendering and illustrates its promise to enhance user experience. Section III derives and validates user experience impairment functions for asymmetric graphics rendering factors introduced in Section II. Section IV validates the network delay impairment function and derives a complete CMG(3D)-UE model. Section V formulates and solves the problem of automatically selecting optimal graphics rendering factors given changing network bandwidth constraints. Section VI shows experimental results with two different games using our CMG(3D) prototype hosted on Amazon Web Service and using commercial cellular network profiles. Section VII proposes future work and concludes the paper.

II. Asymmetric Graphics Rendering

In this paper, 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 (Fig. 1), we can potentially render each view with different texture detail and view distance, leading to *asymmetric graphics rendering*.

Fig. 3(a) 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 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. Fig. 3(b) 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.

We will show in the following why asymmetric graphics rendering is promising to enhance user experience. Figs. 4(a) and 4(b) 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 Fig. 4(b), 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.

III. Impairment Functions Derivation and Validation

In order to learn how graphics rendering factors affect CMG(3D)-UE quantitatively, especially when asymmetric graphics rendering is applied, we define an impairment function I_R as (1).

$$I_R = I_{VD} + I_{TD}. \tag{1}$$

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 (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 [28]. 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.

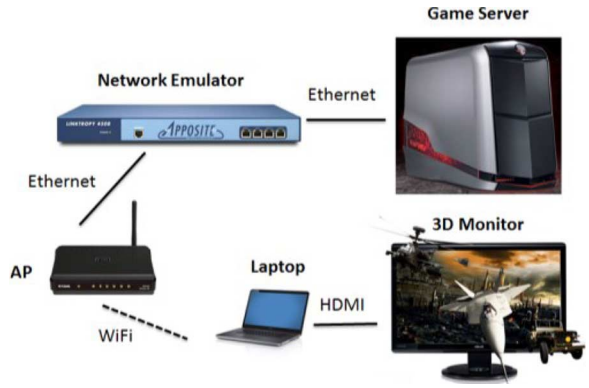

Fig. 5. Testbed for Subjective Experiments.

A. Subjective Experiment Setting

Fig. 5 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 [9]. 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 I 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 II 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

TABLE I
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 II
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

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 [14], [35]. We collect all the evaluations under different graphics rendering factors and use them to derive the impairment functions.

Note that in Table I, 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.

B. 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 I 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 I, 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.

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

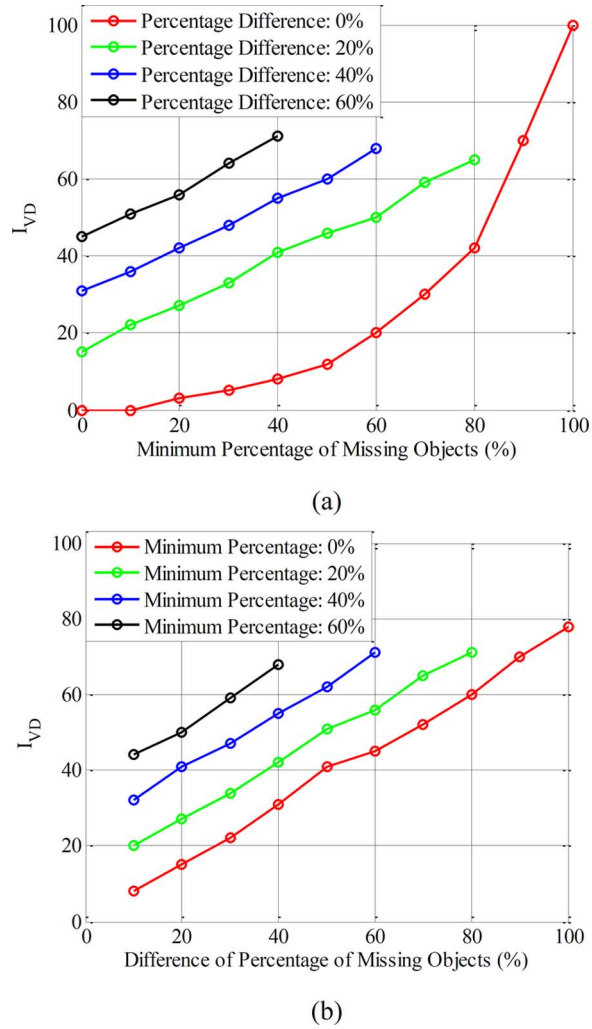

Fig. 6. Subjective test results: (a) top, Relationship between Impairment values and Minimum Percentage of Missing Objects (b) bottom, Relationship between Impairment values and Difference of Percentage of Missing Objects.

are seen by one eye but not the other. Fig. 6(a) shows the relationship between I_{VD} and minimum percentage of missing objects while Fig. 6(b) 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 (2) to describe the relationship when both views have the same percentage of missing objects.

$$I_{VD} = \mathrm{a}P^3 + bP^2 + \mathrm{c}P \tag{2}$$

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 Section IV.

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 (3) shows the relationship. P_{diff} means the difference of percentage of missing objects and $P_{\min}$ means the minimum percentage of missing objects.

$$I_{VD} = eP_{diff} + fP_{\min} + g \tag{3}$$

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 II. 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 III summarizes the results. From Table III 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 (4)) using regression method. Note that in (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.

$$I_{TD} = hTD_{diff}{}^2 + iTD_{diff} + jTD_{\max} \tag{4}$$

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.

C. Impairment Function Verification

In order to verify the functions ((1)– (4) and Table III) derived in the previous subsection, we conducted another set of experiments with a new group of 15 participants (11 male, 4 female; aged $18 \sim 25$), playing the same game, and evaluate using the same criterion (Table II); however, in this set of experiments, the texture detail and percentage of missing objects

TABLE III
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

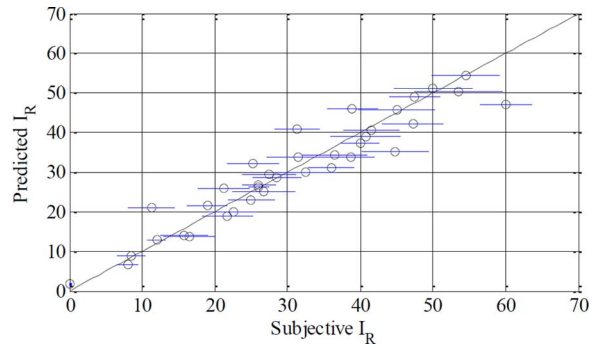

Fig. 7. Validation of I_R with blue line showing 95% confidence interval.

parameters are changed at the same time. Fig. 7 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.

IV. OVERALL USER EXPERIENCE MODELING

In [3]–[5], [21], [28], 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 [28], there are three major categories of objective factors: graphics rendering factors, video encoding factors, and mobile network factors. As discussed in Section III, 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

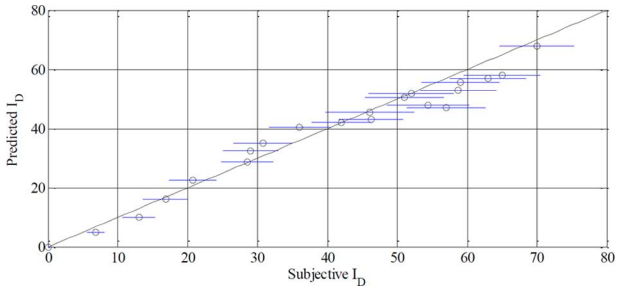

Fig. 8. Relationship between predicted and subjective I_D value with blue line showing 95% confidence interval.

visual quality and response time. In this paper, 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 Section III. 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 [3]. 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).

A. Impairment Function Validation for Network Delay

In our previous work [3], the impairment caused by network delay for 2D cloud gaming has been studied and the impairment function is as follows:

$$I_D = \begin{cases} 0 & (T_1 > Delay > 0) \\ T_0[\frac{(Delay-T_1)}{(T_2-T_1)}] & (T_2 > Delay > T_1) \\ T_0 + \alpha(Delay - T_2) & (Delay > T_2) \end{cases} . \quad (5)$$

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

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 Fig. 5. The value of network delay parameters are shown in Table IV. 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. Fig. 8 shows the relationship between subjective impairment given by people with the objective impairment computed using (5). We also plotted 95% confidence interval for each measurement as blue lines in the figure. The correlation between predicted I_D (y-axis) and subjective I_D (x-axis) is 0.97. This high correlation indicates that although (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.

B. 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.

TABLE IV
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

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 [14]. 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:

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

In (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:

$$R = 100 - I = 100 - I_R - I_D + C_{DR}(\sqrt{I_D \cdot I_R}) \quad (7)$$

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.

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 (Section III-C). 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 V whose scale is mapped to Table II via (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 (6) and (7) to compute our predicted UE score and compare it with the subjective UE score given by the testers to validate the model. Fig. 9(a) 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.

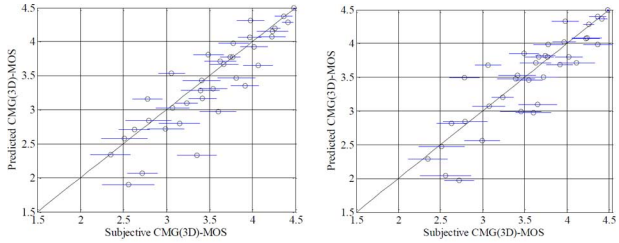

Fig. 9. (a) left, Relationship between predicted MOS and subjective MOS for the game Planeshift, (b) right, Relationship between predicted MOS and subjective MOS for the game Broadsides, Blue line showing 95% confidence interval.

TABLE V
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

To demonstrate the applicability of the CMG(3D)-UE model to other games, we applied it to another 3D game Broadsides [26], 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 VI 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. Fig. 9(b) 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 VI and Fig. 9(b), 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.

V. 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 Fig. 4, 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 [4], [28] 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 follows:

TABLE VI
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	

Given:

Network bandwidth BW

Find:

The optimal graphics rendering factors, including P_L, P_R, TD_L and TD_R, to minimize I

$$I^{opt} = \min I = I_R + I_D - C_{DR}(\sqrt{I_{\mathrm{D}} \cdot I_{\mathrm{R}}}) \tag{8}$$

s.t.

$$\begin{aligned} P_{lower\_bound} \le P_R \le P_L \le P_{upper\_bound} \\ TD_{lower\_bound} \le TD_L \le TD_R \le TD_{upper\_bound} \\ Delay \le D_{Threshold}. \end{aligned} \tag{9}$$

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

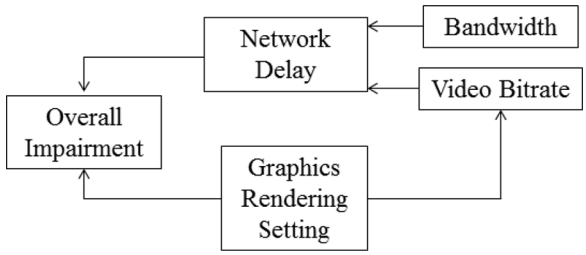

Fig. 10. Factors affecting overall impairments.

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.

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 (8) and Fig. 10, 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 Section V-C 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 D.

The rest of this section is organized as follows. In Section V-A, we propose and verify a model to predict the bitrate of the encoded 3D video resulting from rendering performed with given rendering factors. Section V-B 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. Section V-C proposes the overall algorithm to solve the proposed optimization problem.

A. 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 [22] 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 [23] 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.

$$B_{SUM} = B_L + B_R. \tag{10}$$

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

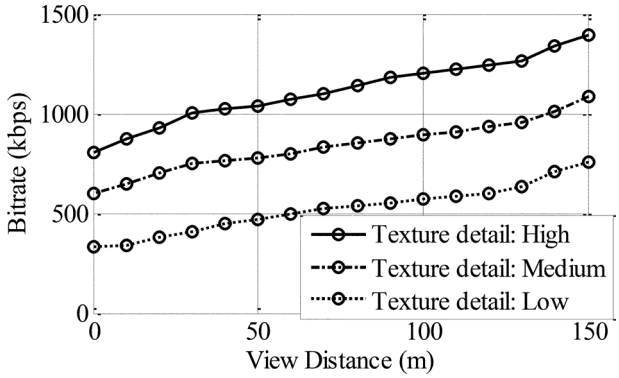

Fig. 11. Derivation of relationship between graphics rendering factors and video bitrate.

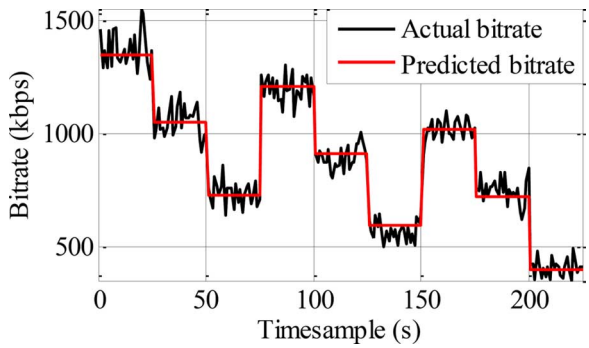

Fig. 12. Verification of the relationship between graphic rendering factors and video bitrate.

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 Fig. 11. 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 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.

$$\begin{aligned} B_L &= kTD_L^2 + lTD_L + mVD_L + n \\ B_R &= kTD_R^2 + lTD_R + mVD_R + n. \end{aligned} \tag{11}$$

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

Fig. 12 shows the validation results. Predicted values (computed using (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.

B. Delay Prediction and Verification

Previous work has been done to model the network and predict delay (like [29], [30]). 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 (12) shows this relationship.

$$D(t) = DN(t) + Ds(t) + Dp(t) \tag{12}$$

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. Equations (13) and (14) show how $D_N(t)$ is calculated.

$$D_N(t) = \max\left(\frac{(S_p(t) + S_c(t))}{BW(t) - T_{INT}}, 0\right) \tag{13}$$

$$S_c(t) = B(t) \cdot T_{INT} \tag{14}$$

$$S_p(t + T_{INT}) = \max(S_p(t) + S_c(t) - BW(t) \cdot T_{INT}, 0) \tag{15}$$

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 (13), it will take $(Sp(t) + S_C(t))/BW(t)$ 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 (15) 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.

We verify the accuracy of the above procedure by conducting the following experiment. Fig. 13 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

Fig. 13. Testbed for measuring delay.

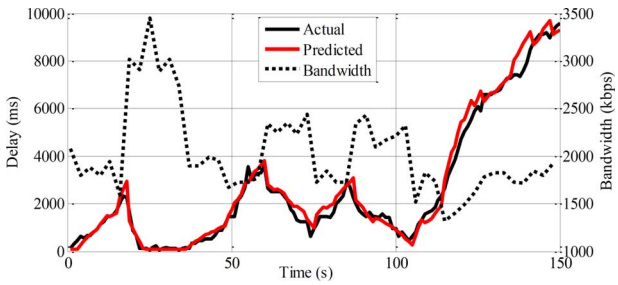

Fig. 14. Results showing actual and predicted delay.

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. Fig. 14 shows the result. Black dotted line is the network bandwidth trace captured in a real 4G-LTE environment. Red solid line shows the predicted delay results calculated by (12)∼(15) 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 (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 Fig. 14 that the predicted results match the actual experimental results very well. Note that though in the above experiment we consider the case where $Ds(t)$ and $Dp(t)$ values are zero, in our CMG(3D) application they will not be. As we discuss later in Section VI, 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 (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 [6] and BART [7] to accurately estimate the network bandwidth by injecting probing traffic. BART [7] 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 [7] requires 0.2 Mbps overhead to achieve reasonable accuracy. In our approach, we use BAR [7] 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.

C. Asymmetric Rendering Optimization Algorithm

From the above sections, it can be seen that by applying the impairment functions ((1)∼(5)), delay prediction model ((12)∼(15)) and relationship between graphics rendering factors and video bitrate ((10)∼(11)), we can expand the optimization function and delay constraint from the problem formulation ((8)) to depend only on the four rendering factors P_1, P_2, TD_1, TD_2 and the current network bandwidth BW. Due to space constraints, we have included the expanded functions in the supplementary material submitted with the paper. 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[24]. We describe next

Algorithm 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)
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
If $I_{temp} < I^{OPT}{}_{temp}$
$I^{OPT}{}_{temp} = I_{temp}$
$P_1^{OPT}{}_{temp} = P_{1temp}$ $P_2^{OPT}{}_{temp} = P_{2temp}$
$TD_1^{OPT}{}_{temp} = TD_{1temp}$ $TD_2^{OPT}{}_{temp} = TD_{2temp}$
Endif
Else if s = Non-integer solution
For *i*=1; *i*<$2^{\text{number\_of\_non\_integer\_values}}$; *i*++
Determine the *i*th variable range for
$subproblem_i$
inqueue($subprobelm_i$, *subproblem_queue*)
Endfor
If s = No feasible solution
continue
Endif
Endwhile
Step 5: Return $I^{OPT} = I^{OPT}{}_{temp}$
$P_1^{OPT} = P_1^{OPT}{}_{temp}$ $P_2^{OPT} = P_2^{OPT}{}_{temp}$
$TD_1^{OPT} = TD_1^{OPT}{}_{temp}$ $TD_2^{OPT} = TD_2^{OPT}{}_{temp}$

Fig. 15. Psuedo-code for ARA Algorithm.

our branch and bound based algorithm Asymmetric Rendering Adaptation (ARA) shown in Fig. 15.

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 (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_{temp}^{OPT}, 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 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 $TD_1 <= 1$. *subproblem2* will also be the same as *subproblem* but with the constraint that $TD_1 >= 2$. 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.

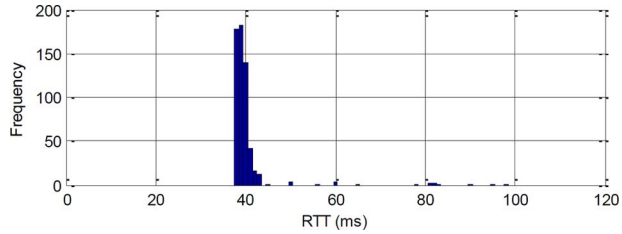

Fig. 16. RTT measured from AWS to test device.

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 V-C), 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.

VI. 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 Fig. 5, 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 in Section III-A, we emulate network traces collected from a 4G-LTE network using the network emulator in the testbed.

Firstly, in order to determine the propagation delay, $D_P(t)$, in (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. Fig. 16 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

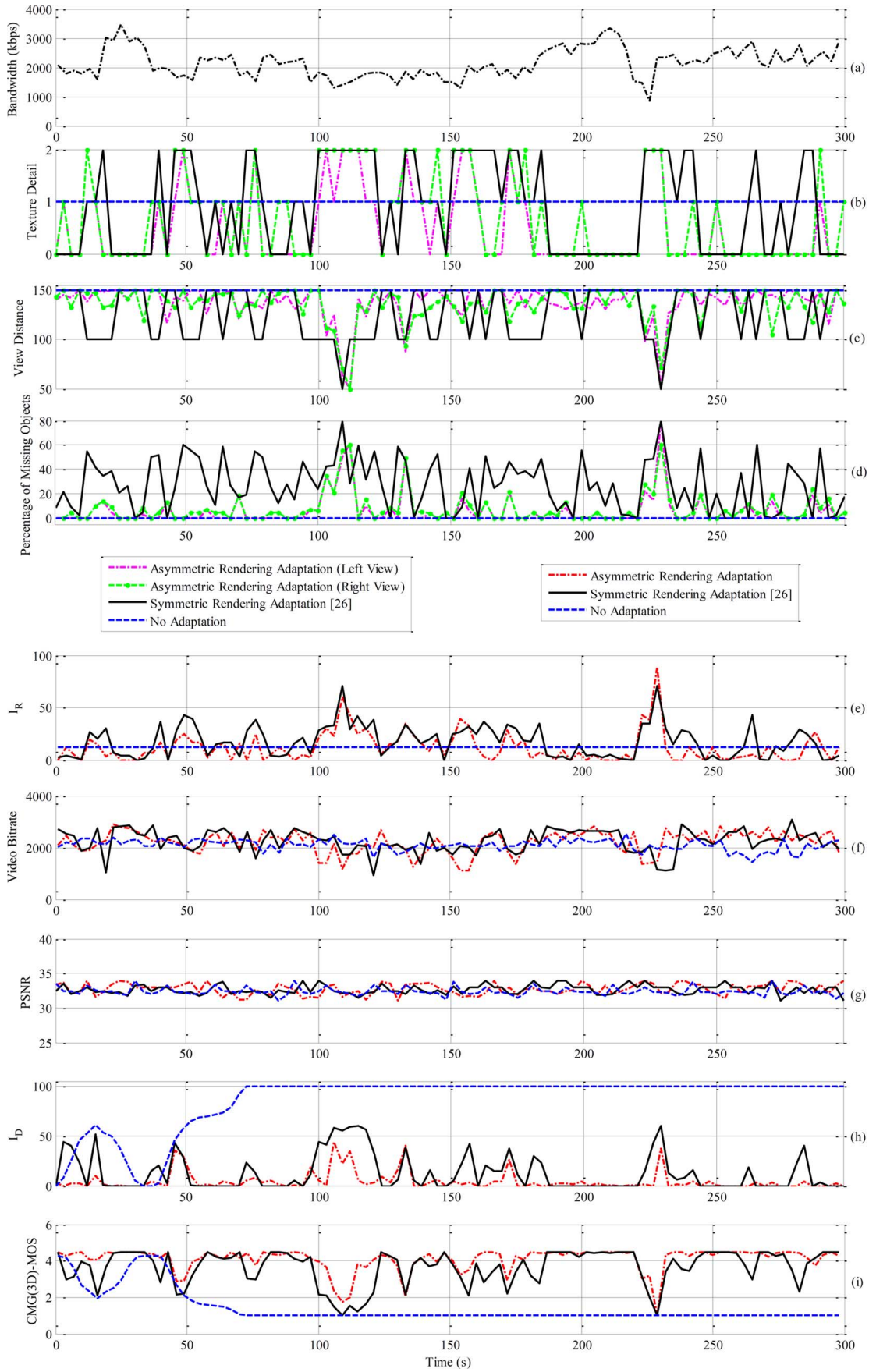

Fig. 17. Results for game PlaneShift.

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 (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 [25] 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. Fig. 17 shows the results for Planeshift and figure 18 shows the results for Broadsides [26]. Due to space constraints, we include figure 18 in the supplementary material submitted with the paper. For both games, we show step by step results in the following order.

Fig. 17(a) and 18(a) 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. Figs. 17(b),17(c),17(d),18(b), 18(c) and 18(d) 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 that ARA can choose better combinations (higher values) of texture detail and view distance than SRA. Fig. 17(e) and 18(e) show the value of rendering impairment, I_R, 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.Figs. 17(f) and 18(f) 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. Figs. 17(g) and 18(g) 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. Figs. 17(h) and 18(h) show I_D which is calculated using actual delay measured. We can see that ARA can maintain low delay much better than the other two strategies.

TABLE VII
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 VIII
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

Fig. 17(i) and 18(i) 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 Fig. 17(a). 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 V at the end of each 1-min segment for each adaptation algorithm. Tables VII and VIII list the minute-wise average subjective CMG(3D)-MOS for the three algorithms with 95% confidence interval range. (For example, "3.56 ± 0.04" 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%.

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.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, 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 [25] and is applicable and effective to 3D games of different genres.

In the future, we would like to extend our research reported in this paper in several directions. We would like to extend the subjective tests by asking subjects to play the games for a much longer time with changes of graphics rendering settings from time to time in order to understand 1) whether asymmetric graphics settings may cause players to have visual discomfort and 2) whether frequent changes of the settings which causes fluctuations of video quality will also be perceived as annoyance and what is the best changing frequency. Furthermore, we will extend our model to take into account video quality impairment, and especially research on a new algorithm which can combine asymmetric video encoding with asymmetric graphics rendering technique together to further enhance user experience under given network constraints. Moreover, instead of using the H.264 AVC for 3D video encoding, we will investigate use of more efficient 3D video codecs such as MVC (multi-view video encoder) [11] or Video Plus Depth Codec [11]. We have done some preliminary studies encoding the 3D videos generated by asymmetric graphics rendering using H.264. We observe that the percentage reductions in bitrate that can be achieved by specific rendering factors are quite similar when using MVC compared to when using AVC. Hence, we believe our proposed asymmetric graphics rendering approach will show significant benefits even when using more efficient 3D video codecs.

Besides using asymmetric graphics rendering (and asymmetric video encoding) to adapt the video bit rate, we would also like to explore in the future the possibility of switching from 3D video to delivering 2D video, as the latter may significantly reduce the bitrate and may be an acceptable or better option for certain games. We would like to enable this option and research how and when this option may benefit user experience. And lastly, while the technique proposed in this paper is mostly suitable for 3D games with high motion, we would like to extend the technique to also work with those 3D games which may have lower motion.

In terms of scalability, we note that the proposed asymmetric graphics rendering technique can be used for other cloud based 3D rendering applications like 3D augmented reality, and expanded to multi-view rendering using multiple virtual cameras for a more immersive experience. We plan to explore the extension of asymmetric rendering in the above directions in the future.

REFERENCES

[1] [Online]. Available: http://www.appannie.com/

[2] [Online]. Available: http://www.slideshare.net/3DTablet/top-5-android-3d-tablet-2013

[3] S. Wang and S. Dey, "Cloud mobile gaming: modeling and measuring user experience in mobile wireless networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev. 16.1*, pp. 10–21, 2012.

[4] S. Wang and S. Dey, "Rendering adaptation to address communication and computation constraints in cloud mobile gaming," in *Proc. IEEE Global Commun. Conf. (GLOBECOM'10)*, Miami, FL, USA, Dec. 2010, pp. 1–6.

[5] Y. Liu, S. Wang, and S. Dey, "Modeling, characterizing, and enhancing user experience in cloud mobile rendering," in *Proc. IEEE Int. Conf. Comput., Netw., Commun. (ICNC'12)*, Maui, HI, USA, Jan. 2012.

[6] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "Pathchirp: Efficient available bandwidth estimation for network paths," in *Proc. Passive Active Meas. Workshop*, Apr. 2003, vol. 4.

[7] S. Ekelin, M. Nilsson, E. Hartikainen, A. Johnsson, J. E. Mangs, B. Melander, and M. Bjorkman, "Real-time measurement of end-to-end available bandwidth using Kalman filtering," in *Proc. 10th IEEE/IFIP Netw. Operat. Manage. Symp. (NOMS '06)*, Apr. 2006, pp. 73–84.

[8] K. T. Chen, Y. C. Chang, P. H. Yseng, C. Y. Huang, and C. L. Lei, "Measuring the latency of cloud gaming systems," in *Proc. 19th ACM Int. Conf. Multimedia (MM'11)*, 2011.

[9] [Online]. Available: http://www.planeshift.it/

[10] S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM'09)*, Honolulu, HI, USA, 2009, pp. 1–7.

[11] A. Smolic, M. Karsten, M. Philipp, K. Peter, and W. Thomas, "An overview of available and emerging 3D video formats and depth enhanced stereo as efficient generic solution," in *Proc. IEEE Picture Coding Symp. (PCS '09)*, 2009, pp. 1–4.

[12] G. Saygili, C. G. Cihat, and A. M. Tekalp, "Evaluation of asymmetric stereo video coding and rate scaling for adaptive 3D video streaming," *IEEE Trans. Broadcasting 57.2*, pp. 593–601, 2011.

[13] S. Valizadeh, A. Maryam, and N. Panos, "Bitrate reduction in asymmetric stereoscopic video with low-pass filtered slices," in *Proc. IEEE Int. Conf. Consumer Electron. (ICCE'12)*, pp. 170–171.

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

[15] [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/authors/1182

[16] A. Ojala and P. Tyrvainen, "Developing cloud business models: A case study on cloud gaming," *IEEE Software, 28.4*, pp. 42–47, 2011.

[17] J. Gaudiosi, "Future of cloud gaming: Industry leaders' thoughts," FC Business Intelligence, 2011.

[18] P. E. Ross, "Cloud computing's killer app: Gaming," *IEEE Spectrum*, vol. 46, no. 3, p. 14, Mar. 2009.

[19] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "An evaluation of QoE in cloud gaming based on subjective tests," in *Proc. IEEE Int. Conf. Innovative Mobile and Internet Services in Ubiquitous Comput. (IMIS'11)*, 2011, pp. 330–335.

[20] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE management for cloud applications," *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 28–36, Apr. 2012.

[21] S. Dey, Y. Liu, S. Wang, and Y. Lu, "Addressing response time of cloud-based mobile applications," in *Proc. ACM 1st Int. Workshop Mobile Cloud Comput. Netw.*, 2013, pp. 3–10.

[22] [Online]. Available: http://journal.iis.sinica.edu.tw/paper/1/130146-2.pdf?cd=52C7BB7E94E9D111F

[23] R. Schreier and A. Rothermel, "Motion adaptive intra refresh for the H. 264 video coding standard," *IEEE Trans. Consumer Electron.*, vol. 52, no. 1, pp. 249–253, Feb. 2006.

[24] M. Bremicker, P. Y. Papalambros, and H. T. Loh, "Solution of mixed-discrete structural optimization problems with a new sequential linearization algorithm," *Comput. Structures 37*, no. 4, pp. 451–461, 1990.

[25] S. Wang and S. Dey, "Adaptive mobile cloud computing to enable rich mobile multimedia applications," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 870–883, Jun. 2013.

[26] Broadsides [Online]. Available: http://cse125.ucsd.edu/cse125/2012/cse125g1/

[27] [Online]. Available: http://www.apposite-tech.com/index.html

[28] Y. Liu, S. Wang, and S. Dey, "Content-aware modeling and enhancing user experience in cloud mobile rendering and streaming," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 4, no. 1, pp. 43–56, Mar. 2014.

[29] A. E. Eckberg, "Approximations for bursty (and smoothed) arrival queueing delays based on generalized peakedness," in *Proc. 11th Int. Teletraffic Congr.*, Kyoto, Japan, 1985.
[30] K. W. Fendick and W. Whitt, "Measurements and approximations to describe the offered traffic and predict the average workload in a single server queue," *Proc. IEEE*, vol. 77, no. 1, pp. 171–194, Jan. 1989.
[31] M. Jarschel, S. Daniel, S. Sven, and H. Tobias, "An evaluation of QoE in cloud gaming based on subjective tests," in *Proc. IEEE Int. Conf. Innovative Mobile Internet Services Ubiquitous Comput. (IMIS'11)*, 2011, pp. 330–335.
[32] B. Vankeirsbilck, D. Verslype, N. Staelens, P. Simoens, C. Develder, P. Demeester, C. Develder, P. Demeester, F. De Turck, and B. Dhoedt, "Platform for real-time subjective assessment of interactive multimedia applications," *Multimedia Tools Applicat.*, pp. 1–27, 2013.
[33] B. Vankeirsbilck, T. Verbelen, D. Verslype, N. Staelens, F. De Turck, P. Demeester, and B. Dhoedt, "Quality of experience driven control of interactive media stream parameters," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM '13)*, pp. 1282–1287.
[34] U. Lampe, R. Hans, and R. Steinmetz, "Will mobile cloud gaming work? findings on latency, energy, and cost," in *Proc. IEEE 2nd Inter. Conf. Mobile Services (MS'13)*, Washington, DC, USA, pp. 960–961.
[35] Int. Telecommun. Union "Bt-500–11: methodology for subjective assessment of the quality of television picture," .

Yao Lu is currently a Ph.D. student at the University of California, San Diego. His research interests include mobile multimedia, computer graphics, video encoding, computer networks, and cloud computing.

Yao Liu is currently a Ph.D. student at the University of California, San Diego. His research interests include mobile multimedia, wireless communication, and mobile cloud computing. His industry experiences include interning at Qualcomm R&D in 2010 and Yahoo Inc. in 2013.

Sujit Dey (SM'03–F'14) received the Ph.D. degree in computer science from Duke University, Durham, NC, USA, in 1991. He is a Professor in the Department of Electrical and Computer Engineering, University of California, San Diego (UCSD), where he heads the Mobile Systems Design Laboratory, which is engaged in developing innovative mobile cloud computing architectures and algorithms, adaptive multimedia and networking techniques, low-energy computing and communication, and reliable system-on-chips, to enable the next-generation of mobile multimedia applications. He is the Director of the UCSD Center for Wireless Communications. He also serves as the Faculty Director of the von Liebig Entrepreneurism Center, and is affiliated with the Qualcomm Institute. He served as the Chief Scientist, Mobile Networks, at Allot Communications from 2012 to 2013. He founded Ortiva Wireless in 2004, where he served as its founding CEO and later as CTO till its acquisition by Allot Communications in 2012. Prior to Ortiva, he served as the Chair of the Advisory Board of Zyray Wireless till its acquisition by Broadcom in 2004. Prior to joining UCSD in 1997, he was a Senior Research Staff Member at the NEC Research Laboratories in Princeton, NJ, USA. He has co-authored over 200 publications, including journal and conference papers, and a book on low-power design. He is the co-inventor of 18 U.S. patents, resulting in multiple technology licensing and commercialization. Dr. Dey has been the recipient of six IEEE/ACM Best Paper awards, and has chaired multiple IEEE conferences and workshops.