Content-Aware Modeling and Enhancing User Experience in Cloud Mobile Rendering and Streaming

Yao Liu, Shaoxuan Wang, and Sujit Dey, *Fellow, IEEE*

***Abstract*—Cloud mobile rendering (CMR), where compute intensive rendering is performed on cloud servers instead of on mobile devices, can be a promising approach to enable rich rendering based multimedia applications on battery and CPU constrained mobile devices. However, since the video rendered in the cloud has to be streamed to the mobile device over a wireless network with fluctuating and constrained bandwidth, the resulting user experience can be impacted. In the work of Wang and Dey, 2013, adaptive rendering was proposed as a solution, wherein multiple rendering factors can be adapted such that the encoding bit rate of the rendered video is compatible with network bandwidth. However, changing the rendering factors may itself have adverse impact on user experience, which has not been studied earlier. Moreover, the impact can vary depending on the content scene and complexity. In this paper, we analyze the impairment of rendering factors on user experience while considering content characteristics, and combine the rendering impairments with impairments due to video encoding factors (like bit rate and frame rate) and network factors (like bandwidth and delay) to propose a content-aware model to measure user experience. We demonstrate the accuracy of the content-aware model through subjective testing. Furthermore, we use the model to: 1) demonstrate and investigate the tradeoff between rendering quality and encoding quality of the resulting video; and 2) derive a content-aware adaptive rendering (CAAR) algorithm which can dynamically adjust the rendering factors depending on current network conditions, so as to obtain an optimal tradeoff between rendering and encoding quality and maximize the overall user experience at any time. By conducting experiments in a real cellular network, we demonstrate that the proposed CAAR algorithm can greatly enhance the user experience of cloud rendered and streamed mobile video.**

***Index Terms*—Adaptive video streaming, cloud computing, mobile network, user experience modeling.**

I. Introduction

WITH the growth and success of mobile applications, there is growing interest in enabling rich multimedia applications, which involve 3-D graphic rendering, like multi-player internet gaming [1], [2], virtual reality and augmented reality [3], [4], on mobile devices. Graphic rendering is not only very computation intensive; it can also impose severe challenges on the limited battery capability of an always-on mobile device. While mobile devices are expected to have increased graphics processing capabilities in the future, one can expect a growing gap between the requirements of the emerging 3-D rendering based applications, and the capabilities of mobile devices [5], [6]. Hence, it may be promising to investigate an alternative approach, cloud mobile rendering (CMR), where compute intensive rendering is performed on cloud servers instead of on mobile devices, and the rendered video is encoded and streamed to the mobile devices. Fig. 1 shows the overall architecture of the CMR system. When a client starts a CMR session from a mobile device, a CMR server will initialize a rendering engine and encoder/streamer for this mobile device. Then the CMR server starts to process the application data to render raw video, which is encoded and streamed to the mobile client through the wireless network. On the other hand, control commands from the mobile client are sent from the mobile device to the CMR server.

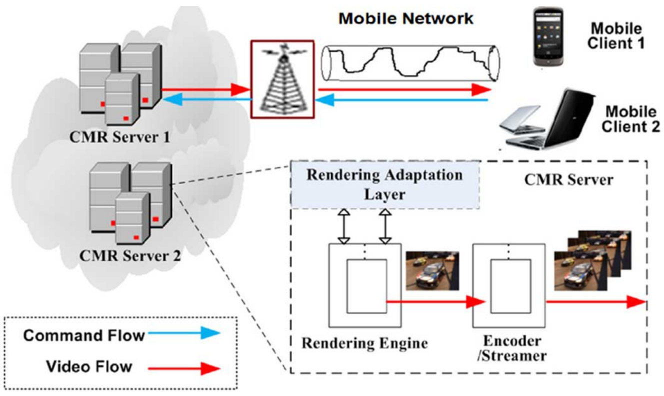


Fig. 1. Overview of CMR system.

Compared with enabling rendering based applications where the rendering is performed on the mobile device itself, the CMR approach has several advantages. Firstly, it has high scalability across various mobile platforms. Take cloud-based mobile gaming as an example. Different versions of the same game will not need to be developed to accommodate the differences among various mobile platforms. Secondly, a CMR based application will be able to use the most advanced rendering technologies, without concern about computation constraints, and hence be able to provide the richest graphic rendering quality. Thirdly, CMR based applications will have significantly less battery consumption than if rendering is performed

Manuscript received September 02, 2013; revised December 14, 2013; accepted December 28, 2013. Date of publication January 30, 2014; date of current version March 07, 2014. This work was supported in part by the National Science Foundation under Grant CCF-1160832. This paper was recommended by Guest Editor C. W. Chen.

The authors are with Mobile System Design Lab, Department of Electrical and Computer Engineering, University of California-San Diego, La Jolla, CA 92093 USA (e-mail: yal019@ece.ucsd.edu; shaoxuan@ece.ucsd.ed; 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/JETCAS.2014.2298921

on the mobile device itself, thus making such rendering applications feasible and popular on mobile devices.

Though CMR can be a promising approach to provide rich rendering multimedia applications for mobile devices, ensuring user experience can be challenging [5], considering 1) rendered video will have to be streamed from the CMR servers to the mobile devices through error-prone wireless networks, and 2) streaming video for each user from CMR server to his/her device will consume significant backhaul and wireless network bandwidth. Motivated by these challenges, we proposed a rendering adaptation technique [6] for cloud mobile gaming (CMG) (a type of CMR application), wherein multiple rendering factors can be adapted such that the bit rate of the encoded rendered video is in accordance with the available network bandwidth, thus minimizing user experience artifacts due to wireless network factors like bandwidth availability, delay, and packet loss. Though adapting rendering factors can mitigate network artifacts, it may also have adverse impact on overall user experience. Therefore, it is vital to derive a user experience model, which can be used to quantitatively measure user perceived experience under different rendering factors. Such user experience model will also help us to determine when and how to adapt rendering factors to mitigate network artifacts, such that the user experience is maximized.

We introduce three categories of factors that will affect the user experience of CMR applications: rendering factors, encoding factors, and network factors. The effects of encoding factors and network factors have been investigated in our previous work [7]. In this paper, we will first study the effect of rendering factors on user experience (UE), and then combine these three types of factors to derive a complete CMR-UE model for video rendered and streamed from cloud servers. To the best of our knowledge, this is the first work to study how rendering of video will affect user experience, and the first UE model suitable for CMR applications where the video rendering factors are adapted. We have also observed that the impairment caused by rendering is highly dependent on the content information of the rendered video, such as what kind of scene and how many 3-D objects are being rendered. Therefore, in the CMR-UE model, we have incorporated video content information in order to further improve modeling accuracy. The proposed content-aware UE model can simultaneously leverage the information of how the scenes are composed and the knowledge of how a certain rendering setting impacts the user experience in different scenes, and therefore can provide a higher accuracy in determining user experience.

Besides being able to quantitatively determine user experience of the rendered and streamed video, another major contribution is to be able to decide the optimal rendering factors and encoding factors such that the CMR user experience is maximized. As will be explained later in Section V, given a certain available network bandwidth, and hence the bit rate budget of the rendered and encoded video stream, there is a tradeoff relation between rendering richness and video quality achieved, increasing rendering richness can lead to lower video quality, and vice versa. In this paper, for the first time, we investigate this tradeoff, and utilize the CMR-UE model as a tool to determine the optimal rendering richness and video quality so as to maximize the overall user experience. Furthermore, we propose a content-aware adaptive rendering (CAAR) algorithm to dynamically select the optimal rendering factor values according to the network condition such that the user experience (as measured by the CMR-UE model) can be maximized at any time. We will show the effectiveness of the proposed CAAR algorithm through experiments using 3.5 G mobile network.

While the proposed approaches (including content aware user experience modeling and content aware rendering adaptation) developed in this article can be useful for any CMR applications, we use CMG (one of the most compute intensive and mobile bandwidth demanding CMR applications) as a running example to introduce and validate our approaches in details.

Several approaches have been developed to model the quality of experience of video streaming [8]–[10], videoconference [11], [12], cloud gaming [13]–[15], and augmented reality [16]. According to these models, user experience is dependent on video quality factors like peak signal-to-noise ratio (PSNR), and network factors like delay, response time, and packet loss. However, these models are not suitable to model the quality of experience of CMR applications with rendering adaptation technique, since they have not considered the impact of graphic rendering factors on user experience. There have been some other literature which has characterized the impact of graphic rendering on processing load [17], [18], but the effect of graphic rendering on user experience has not been studied. Our earlier work [6] proposed how to adapt rendering factors to avoid network congestion and maximize video quality. However, the adaptive rendering algorithm in [6] did not consider the impairment on user experience caused by rendering adaptation itself. Unlike [6], in this paper we propose a new adaptive rendering algorithm which aims at maximizing the overall user experience as measured by the new CMR-UE model proposed here, including video quality as well as rendering richness. Moreover, we show that by being content-aware, the proposed adaptation algorithm can achieve significantly higher user experience.

The remainder of this paper is organized as follows. In Section II, we introduce the factors that affect the user experience of CMR applications, specifically rendering factors that can be used to affect the bit rate of the rendered video but also impair user experience. Section III describes the method to quantitatively measure the impairment of these factors on user experience, taking into account the dependence of the impairments on content. In Section IV, we first introduce CMR-MOS metric as a way to measure CMR-UE, and then derive the content aware CMR-UE model and validate it using subjective testing. Section V describes the proposed content aware adaptive rendering (CAAR) algorithm for efficient streaming of CMR video over wireless networks. Section VI presents the results of experiments conducted with a commercial mobile network, demonstrating the benefit of CAAR algorithm. Section VII concludes this paper.

II. Factors Affecting User Experience of Cloud Mobile Rendering Application

The first step to study and implement the CMR-UE model is to identify the impairment factors, which impacts user ex-

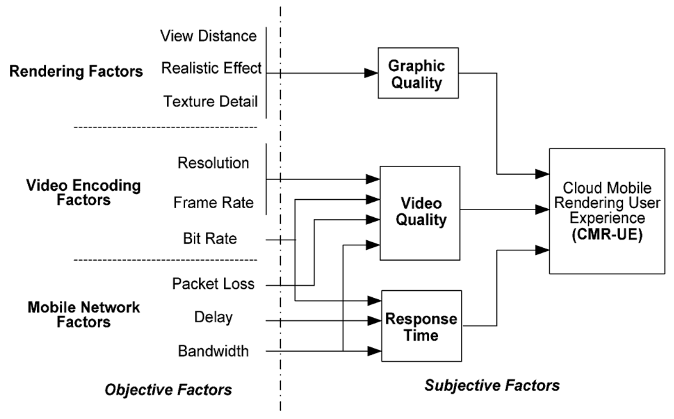


Fig. 2. Objective and subjective factors affecting user experience of CMR applications.

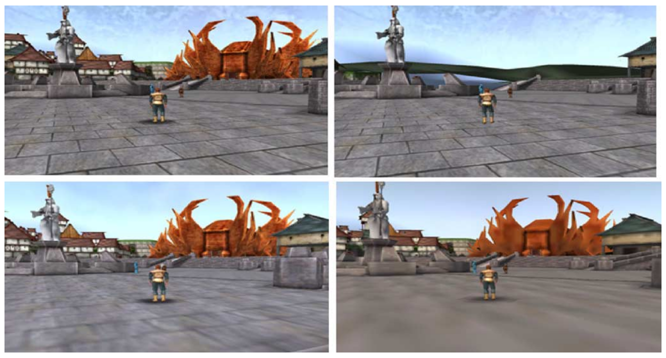


Fig. 3. Graphic quality difference of different settings of *view distance* and *texture detail*: (a) top-left, 300 m view and high texture detail level; (b) top-right, 60 m view and high texture detail level; (c) bottom-left, 300 m view and medium texture detail level; (d) bottom-right, 300 m view and low texture detail level.

perience of a CMR session. In this section, we first introduce the impairment factors of CMR application, and then provide a detailed explanation of how rendering factors affect user experience.

As described in Section I (Fig. 1), during a CMR session, graphic rendering is performed at a cloud server, with the resulting video subsequently encoded and streamed over the bandwidth constrained and error-prone wireless network. Therefore, as shown in Fig. 2, there are three major categories of objective factors: rendering factors, video encoding factors, and mobile network factors. The user experience is determined by three subjective factors: graphic quality, video quality, and response time. Response time is the duration between the user issuing a command and the effect of this command being displayed on the mobile device. It has significant impact on user experience due to the interactivity nature of CMR applications. As shown in Fig. 2, the objective factors would affect the subjective factors in different ways. Rendering factors will only affect the user perceived visual quality of the graphics, while encoding factors and network factors affect user perceived video quality and response time in a complex manner. For instance, video encoding bit rate will affect user perceived video quality as well as the response time since it will determine the amount of video traffic need to be transmitted. Note both the time taken for rendering and encoding also contribute to response time. However, assuming scheduling of CMR applications to the CMR servers are done such that there is no over-utilization of the servers, the rendering and encoding time for a certain CMR application can be expected to be fixed, and hence we do not consider them any more in the discussion below.

We next describe the impact of different rendering factors (Fig. 2) on bit rate of the rendered video as well as the user experience. Fig. 3 shows graphic quality differences when different settings are used for different rendering factors in a CMG application. Fig. 3(a) and (b) compares the graphic quality between two different view distance settings (300 m and 60 m). Fig. 3(a)–(c) compares the graphic quality for different rendering texture detail levels. A shorter view distance [Fig. 3(b)] or a lower texture detail level [Fig. 3(c) and (d)] can have significantly less bit rate needed to encode the rendered video. For instance, to achieve a PSNR value of 32 dB, the required video bit rates of the rendered video shown in Fig. 3(a)–(d) are about 700 kb/s, 590 kb/s, 530 kb/s, and 420 kb/s, respectively. We can leverage these characteristics and adapt the graphic quality (rendering richness) and therefore adapt the bit rate of the streamed video to avoid network congestion. However, obviously shorter view distance or lower texture detail level has impairments on user perceived gaming quality. Therefore, we need to study how view distance or texture detail level affects user experience. The above understanding will help guide the CMR system to perform the right tradeoff between increasing view distance/texture detail and possibly affecting graphic visual quality, versus having network congestion affecting video quality and response time. Motivated by this, one of the major contributions of this paper is to develop (Sections III and IV) a user experience model (CMR-UE), which takes into account rendering factors in addition to network and video encoding factors [7], to quantitatively measure the user perceived CMR service quality.

III. Derivation and Validation of Impairment Functions for Rendering Factors

In this section, we focus on studying the effects of rendering factors on CMR-UE. Based on a subjective study, we derive an impairment function $\mathrm{I_R}$ which denotes the impairments caused by graphic rendering factors as discussed in Section II (Fig. 2). In the next section (Section IV), we will derive a complete CMR-UE model with this new $\mathrm{I_R}$, and video encoding impairment function ($\mathrm{I_E}$) and network impairment functions ($\mathrm{I_D}$ and $\mathrm{I_L}$) which were derived in [7].

A. Impairment Function for Each Rendering Factor

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

As shown in Fig. 2, we consider the effect of three common graphic rendering factors, *View Distance*, *Realistic Effect*, and *Texture Detail*, on the graphic quality of CMR applications. Each of these factors would affect graphic quality from a different aspect, and the total rendering impairment $\mathrm{I_R}$ can be formulated with three impairment sub-functions

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

TABLE I
GRAPHIC QUALITY RATING CRITERIA AND I_R VALUES

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

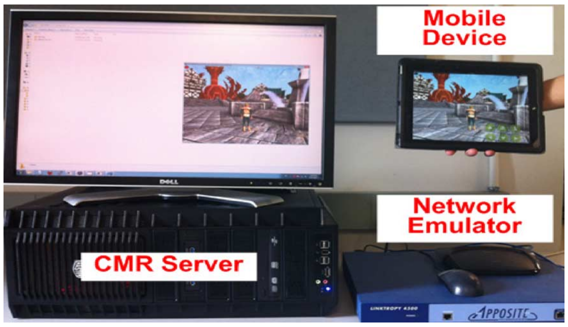


Fig. 4. Experiment framework.

TABLE II
PARAMETERS FOR SUBJECTIVE EXPERIMENT

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

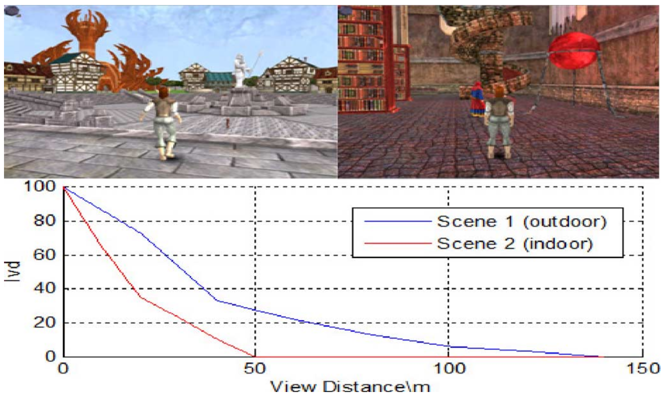


Fig. 5. Relationship between I_{VD} and view distance value for different game scenes: (a) top left, screenshot of scene 1; (b) top right, screenshot of scene 2; (c) bottom, relationship between subjects' evaluation IVD and view distance.

where I_{VD} represents the impairment caused by reducing view distance, I_{RE} indicates the impairment caused by reducing realistic effect, and I_{TD} indicates the impairment of reducing texture detail.

In the next subsections (Sections III-B and III-C), we derive the impairment sub-functions I_{VD}, I_{RE}, and I_{TD} by conducting a set of subjective quality assessment experiments, and the accuracy of the linear model [(1)] will be validated with another set of subjective experiments in Section III-D.

B. Subjective Quality Assessment Experiment

To derive the impairment sub-functions (I_{VD}, I_{RE}, and I_{TD}) in (1), we developed and conducted a subjective quality assessment experiment, using a group of participants comprising of 18 students at UCSD. We used CMG application as a demonstration example of CMR application. We use a MMORPG game called PlaneShift [19]. The experiment environment is shown in Fig. 4, where the mobile device is connected to the game server through a network emulator, which can be used to control the network conditions, such as delay, packet loss pattern, bandwidth, etc. In the experiment environment, the game logic and rendering tasks are executed by the server, while the mobile device sends control commands and displays gaming video. We control the network emulator to keep the network bandwidth to be sufficiently large, and only change the rendering factors on the server. Each experiment participant plays the game on the mobile device with different combinations of the rendering factors shown in Table II, and provided assessment of the rendering impairments IR using an impairment rating system shown in Table I. Finally, the results of the study group were tabulated for further analysis and derivation of impairment functions.

C. Derivation of Impairment Function I_{VD}, I_{RE}, and I_{TD}

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

In order to derive the impairment due to view distance, I_{VD}, we first investigate the relation between the average impairment evaluation by the participants and the view distance values. We find that this relation is very different for different game scenes. For example, Fig. 5 depicts the players' average evaluation of impairment for different view distances, in the case of two game scenes. Fig. 5(a) is a screenshot of scene 1, which is a representative outdoor game scene. Fig. 5(b) is a screenshot of scene 2 which is an indoor scene. For each scene, we plot the average evaluations of I_{VD} for different view distance values, as shown in Fig. 5(c). Based on the experiment results, it is obvious that the relationship between I_{VD} and view distance value are very different for different game scenes. In other words, the impairment of view distance, I_{VD}, is not solely a function of view distance value, but it is also related with the game scene (content) rendered. Since during a gaming session, numerous possible game scenes may be rendered depending on the actions of

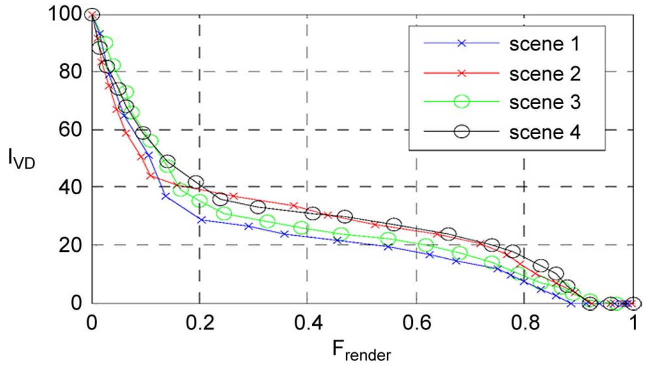


Fig. 6. Relation between subjects' evaluation $\mathrm{I_{VD}}$ and $\mathrm{F_{render}}$.

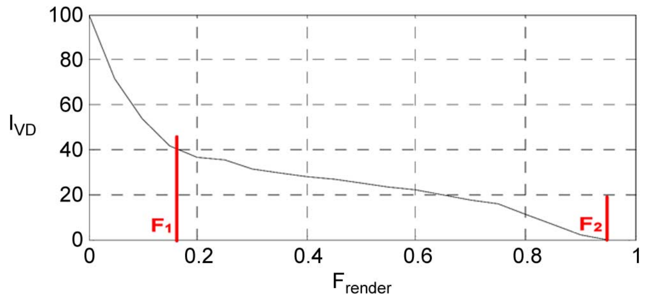


Fig. 7. Relation between subjects' average evaluation $\mathrm{I_{VD}}$ and $\mathrm{F_{render}}$.

the gamers, it will be challenging, if not impossible, to develop and use a model for $\mathrm{I_{VD}}$ which depends on the specific scenes.

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

Given a view distance value, the corresponding F_{render} value can be computed conveniently by exploiting the video content information. In CMR applications, some video content information can be extracted from the graphic engine during the rendering process. For example, we can extract the location information of all the 3-D objects (avatars, buildings, trees, etc.) in the game scenes [shown as Fig. 5(a) and (b)], as well as the distance values from these 3-D objects to the camera. Therefore, for a view distance value *dist*, we can check how many 3-D objects in the game scene have a smaller distance than *dist*. Then F_{render} can be calculated by dividing this number by the total number of 3-D objects.

Having defined F_{render}, we conduct the same experiment process as described in Section III-B to derive $\mathrm{I_{VD}}$ using metric F_{render}. As we vary view distance, we collect subjects' evaluations about $\mathrm{I_{VD}}$ under 20 different game scenes. This time we record subjects' evaluations together with corresponding F_{render} values. Among the 20 game scenes, we plot in Fig. 6 the relation between subjects' average evaluation of $\mathrm{I_{VD}}$ and corresponding F_{render} for four representative scenes. We can see that although the relationship between subjects' evaluation of $\mathrm{I_{VD}}$ and F_{render} are not identical for these four different game scenes (two outdoor scenes and two indoor scenes), their trends are very similar, and we can observe similar trend for the other 16 game scenes. This result validates our idea that by formulating $\mathrm{I_{VD}}$ as a function of F_{render} can make $\mathrm{I_{VD}}$ scene-independent.

TABLE III
IMPAIRMENT FUNCTION FOR REALISTIC EFFECT

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

To derive this scene-independent impairment function $\mathrm{I_{VD}}$, for each F_{render} value, we compute the average $\mathrm{I_{VD}}$ score over all the 20 game scenes. The resulting average curve is plotted in Fig. 7. From Fig. 7, we have two observations: first, there is no impairment when F_{render} is bigger than a certain threshold $\mathrm{F_2}$ (for this specific game PlaneShift, $\mathrm{F_2} = 0.95$); second, there is an obvious slope change at the point where F_{render} equals a certain value $\mathrm{F_1}$ ($\mathrm{F_1} = 0.15$ for this game). $\mathrm{F_1}$ and $\mathrm{F_2}$ divide the view distance value into different segments. Consequently, the impairment sub-function $\mathrm{I_{VD}}$ can be derived, using linear regression analysis, as the following:

$$\mathrm{I_{vd}} = \begin{cases} 0, & (F_{\mathrm{render}} > \mathrm{F_2}) \\ 40^* \left[\frac{(\mathrm{F_2} - F_{\mathrm{render}})}{(\mathrm{F_2} - \mathrm{F_1})}\right], & (\mathrm{F_2} > F_{\mathrm{render}} > \mathrm{F_1}) \\ \alpha^*(\mathrm{F_1} - F_{\mathrm{render}}) + 40, & (\mathrm{F_1} > F_{\mathrm{render}}) \end{cases}. \tag{2}$$

In (2), coefficient $\mathrm{F_1}$ equals 0.15, $\mathrm{F_2}$ equals 0.95 and α is 30, respectively.

We use a similar method to derive the impairment functions for the other rendering parameters, realistic effect, $\mathrm{I_{RE}}$, and texture detail, $\mathrm{I_{TD}}$. We vary only texture detail or realistic effect value according to Table II, while keeping the other two rendering factors at the highest value. Then we use the average subjective score over all the subjects and game scenes as the results for $\mathrm{I_{RE}}$ or $\mathrm{I_{TD}}$, which are listed in Table III and IV, respectively. Unlike $\mathrm{I_{VD}}$, the results of $\mathrm{I_{TD}}$ and $\mathrm{I_{RE}}$ are given in table instead of equation, because there are only three options for texture detail and realistic effect. The standard deviation of $\mathrm{I_{RE}}$ and $\mathrm{I_{TD}}$ scores over different game scenes is 0.56 and 1.89, respectively. These small standard deviation values imply that both $\mathrm{I_{RE}}$ and $\mathrm{I_{TD}}$ values have very weak dependencies on game scene/content.

Derivation of impairment sub-functions $\mathrm{I_{VD}}$, $\mathrm{I_{RE}}$, and $\mathrm{I_{TD}}$, allows derivation of the impairment function $\mathrm{I_R}$ according to (1). Note that the values shown in Tables III and IV are specifically for Planeshift, and will be different for other applications; however, the methodology to derive them will be same. Next,

TABLE IV
IMPAIRMENT FUNCTION OF TEXTURE DETAIL

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

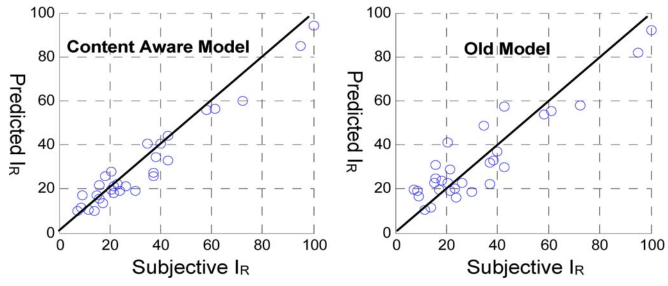


Fig. 8. Relationship between predicted and subjective I_R value: (a) left: new model which include video content information; (b) right: old model which does not consider video content information.

we will validate the accuracy of I_R using another set of subjective assessment experiments.

D. Validation of Impairment Function I_R

To validate impairment function I_R, we conducted another set of experiments with a new set of participants, where the participants now provide assessment of rendering impairment I_R when multiple rendering factors, I_{VD}, I_{RE}, and I_{TD} are varied simultaneously. For each such experiment, we compare the subjective I_R scores the participants give and the predicted I_R values calculated from our impairment function I_R [(1) and (2), Tables III and IV]. We plot the results in Fig. 8(a). For comparison, we also plot in Fig. 8(b) the subjective I_R scores with I_R values calculated by a preliminary model developed in [21], which does not take into account video content information. From Fig. 8, we see that the correlation between predicted I_R (y-axis) and subjective I_R (x-axis) is 0.92 using the proposed content aware I_R model, and 0.85 using the I_R model that does not incorporate video content information. This result demonstrates that I_R can predict with high accuracy the total impairment of rendering factors on user experience.

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

IV. Derivation of Cloud Mobile Rendering User Experience Model

In this section, we develop a CMR-UE model, incorporating the impairment due to rendering factors (developed in Section III) in addition to impairments due to video encoding and wireless network factors that we had developed in [7]. We present results of subjective experiments conducted to derive and validate the CMR-UE model.

We define CMR mean opinion score (CMR-MOS) as a measurement metric for CMR-UE. Since CMR-MOS is determined by rendering, encoding and network factors as shown in Fig. 2, we attempt to formulate it using the impairment functions of these factors, similar to the framework of ITU-T E-Model [20]. Although the ITU-T E-model is originally developed for audio transmission applications, the framework of transmission rating factor R, which represents the overall user experience, can also be applied in our study. The function of CMR-MOS formulated by R factor can be found in the ITU-T E-model. We duplicate that function for our CMR-MOS formulation

$$\mathrm{CMR-MOS} = 1 + 0.035\mathrm{R} + 7\times 10^{-6}\mathrm{R}(\mathrm{R}-60)(100-\mathrm{R}). \tag{3}$$

In (3), the transmission rating factor R takes value from range [0, 100] (the higher R, the better CMR-UE). CMR-MOS is related with R through nonlinear mapping, and it is within the range of [1, 4.5].

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

$$\mathrm{R} = 100 - \mathrm{I_E} - \mathrm{I_D} - \mathrm{I_L} - \mathrm{I_R} + \sum_{\substack{i,j\in\{\mathrm{E,D,I,R}\}\\ i\neq j}} \mathrm{f_{ij}(I_i, I_j)}(\mathrm{R} > 0). \tag{4}$$

In (4), R is composed of impairment functions for video encoding, $\mathrm{I_E}$, network delay, $\mathrm{I_D}$, network packet loss, $\mathrm{I_L}$ and graphic rendering, $\mathrm{I_R}$. The impairment functions $\mathrm{I_E}$, $\mathrm{I_D}$, and $\mathrm{I_L}$ have been derived in our previous work [7]. Function $\mathrm{f_{ij}(I_i, I_j)}$ indicates the cross-effect of two different impairments and is used to compensate and adjust the R factor, because when several impairments happen simultaneously, the overall impairment will be different from the sum of each impairment.

In order to derive the formula of function $\mathrm{f_{ij}(I_i, I_j)}$, and also validate the accuracy of the overall CMR-UE model, we conduct another set of subjective quality assessment experiment with a new group of participants. Totally 23 volunteers from UCSD are selected for this round of experiments. The same experiment framework (as described in Section III) is utilized, but this time we change multiple factors (encoding, rendering, and network factors) simultaneously and ask participants to evaluate their experience. The encoding factors (frame rate, video bit rate, etc.) and rendering factors (view distance, realistic effect, etc.) are varied at the game server, and the network factors are controlled by the network emulator. Each experiment participant played the game with different encoding, rendering and network factors and gave evaluation on his/her user experience using a CMR-MOS score rating system shown in Table V.

During the subjective experiments, we have collected the participants' evaluations for different combinations of encoding, rendering, and network factors. For each combination of these factors, we compute the average evaluation among the participants, and use this average value as the subjective evaluation of user experience for that combination of factors. The subjective evaluations for all different combinations of factors form a set of test results. Next, we randomly select 60% of the test results, and use them to train the model for R [(4)]. During the

TABLE V
CMR-MOS RATING AND "R" VALUES

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

TABLE VI
COMPENSATION COEFFICIENTS IN (5) FOR GAME PLANESHIFT

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

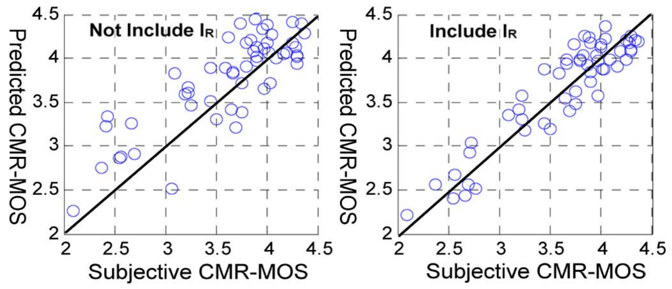


Fig. 9. Relation between predicted and subjective CMR-MOS for game PlaneShift: (a) left, old model, not including rendering impairment; (b) right, proposed model, including rendering impairment.

training, we use different types of functions for $f_{ij}(I_i, I_j)$, including $(I_i + a \times I_j)^n$, $I_i^n \times I_j^m$, and $e^{(I_i + a\times I_j)^n}$, and use linear regression to compute the coefficients for the functions. Then we use the other 40% of test results to validate the proposed R model with all possible $f_{ij}(I_i, I_j)$ functions. Finally, we select the function shown in (5), since it achieves the highest correlation result in the model validation process

$$f_{ij}(I_i, I_j) = C_{ij}\sqrt{I_i{}^{*}I_j}. \quad (5)$$

The C_{ij} coefficients can be determined by performing regression analysis using values of R calculated using (4), (5) and results of subjective tests. The results are shown in Table VI.

Having derived the complete CMR-UE model [(3)–(5) and Table VI], we now use the rest of the subjective tests to validate the model. Fig. 9(b) shows the correlation between predicted CMR-MOS scores computed by the CMR-UE model (y-axis) and subjective CMR-MOS scores (x-axis) for each of the tests. From Fig. 9(b), we observe a high correlation of 0.90 and hence conclude the proposed model can accurately predict user experience. For comparison purpose, in Fig. 9(a) we also show a set of correlation results using the old UE model we proposed in [7] for CMG, which does not consider the rendering impairment I_R. The correlation value for Fig. 9(a) is 0.83. We conclude that the new CMR-UE model which includes I_R leads to better correlation with people's subjective evaluation.

TABLE VII
COMPENSATION COEFFICIENTS IN (5) FOR GAME BROADSIDES

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

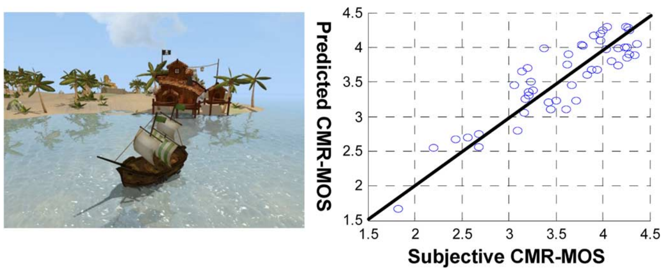


Fig. 10. Evaluating CMR-UE model on second game Broadsides [22] (a) left, screenshot of game Broadsides; (b) right relation between predicted and subjective CMR-MOS.

To demonstrate the applicability of the CMR-UE model on other games, we applied it to another 3-D game Broadsides [22], which belongs to a different genre than game PlaneShift. While PlaneShift is a role playing game, Broadsides is a first person shooter game. For the new game, we apply the same approach for validating the CMR-UE model. We conduct subjective tests and collect evaluation scores under different combinations of the rendering, encoding and network factors. We use 60% of test results for training the model of R and computing the coefficient values, and use the other 40% of test results for validating the CMR-UE model. Table VII shows the values of coefficients C_{ij} for game Broadsides. Comparing these C_{ij} values with the coefficients for game PlaneShift (Table VI), we can see that for different games, the coefficients C_{ij} are different. Fig. 10(a) is a screenshot of the game Broadsides, and Fig. 10(b) is the scatter plot of the relationship between subjective and predicted CMR-MOS. For the new game Broadsides, the correlation is 0.88. From Table VII and Fig. 10(b), we can see that for the new game, after training and refitting the coefficients C_{ij}, the proposed CMR-UE model will also lead to high modeling accuracy.

Note that although for each CMR application (more specifically for each 3-D game) the coefficients C_{ij} need to be derived through the subjective assessment experiments, we still regard the proposed CMR-UE model to be generally applicable because: 1) although the coefficients C_{ij} are different, the framework of CMR-UE model [(3)–(5)] can be applied for different games and have high modeling accuracy according to our experiment results; 2) the amount of work of this subjective experiment is manageable, considering the amount of work and money spent on developing a typical 3-D game; 3) the coefficients C_{ij} need to be trained only once for each game and can be used forever.

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

V. Content-Aware Adaptive Rendering Algorithm

Mobile networks are characterized by unpredictable and sometimes fast bandwidth fluctuations, which can seriously affect the rendered video streamed from the CMR servers to the mobile device, in terms of packet loss and latency. An effective way to cope with the throughput fluctuations of mobile networks is to adapt the rendering factors so that the required encoding bit rate for the resulting rendered video can be adapted according to available network bandwidth [6]. However, as discussed in Section II and illustrated in Fig. 3, while lowering rendering settings can reduce the encoding bit rate needed to achieve a certain video quality (PSNR), it can also negatively affect graphics quality, and hence the overall user experience, which was not considered in [6]. In other words, given a certain available network bandwidth, and hence a fixed video encoding bit rate budget, there is a tradeoff between rendering richness (graphics quality) achieved by graphics rendering and the video quality achieved by the video encoder. The question that needs to be addressed is how to select the optimal rendering settings such that the overall user experience is maximized, when taking into consideration both graphics quality and video quality? Fortunately, the proposed CMR-UE model (developed in Section IV) opens up the possibility to answer this question.

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

This rest of this section is organized as follows. In Section V-A, we explain the tradeoff between rendering richness and video frame quality under a fixed bit rate budget, which explains why rendering complexity need to be adapted. In Section V-B, we give an overview of proposed CAAR algorithm. In Section V-C, we introduce the offline step. Sections V-D and V-E explain in detail the two online steps. The effectiveness of the CAAR algorithm will be shown in the next section.

A. Tradeoff Between Rendering Richness and Video Frame Quality Under a Fixed Bit Rate Budget

Given a certain available network bandwidth, the optimal encoding setting would be using constant bit rate (CBR) encoding to make sure the video bit rate is fixed and below the available network bandwidth to avoid network congestion. For a fixed video bit rate, if we reduce the rendering complexity for instance lowering view distance (impairment of rendering $\mathrm{I_R}$ will increase), the content complexity in each video frame will reduce. Thereby the video encoder will have less compression on each video frame, leading to a higher video quality (impairment of encoding $\mathrm{I_E}$ will decrease). Fig. 11 shows the video quality difference using two different rendering settings (view distances are 150 m and 70 m) but keeping the same encoding setting (bit rate target is 300 kb/s and frame rate is 12 fps). It is obvious that reducing the view distance from 150 m (left figure) to 70 m (right figure) will provide significant improvement in video frame quality while keeping the same video bit rate (the left figure looks much more blurred). This is because the content complexity (the number of 3-D objects) in the left figure where the view distance is 150 m is much larger than the right figure where the view distance is 70 m. Although reducing the content complexity by lowering rendering setting can improve video frame quality, it has negative impact on graphics quality and hence overall user experience. Therefore, to decide the optimal rendering settings which can maximize CMR-UE, we have to investigate the tradeoff between rendering richness (inverse to $\mathrm{I_R}$) and video frame quality (inverse to $\mathrm{I_E}$) on CMR-UE, which will be discussed in detail in the next subsections.

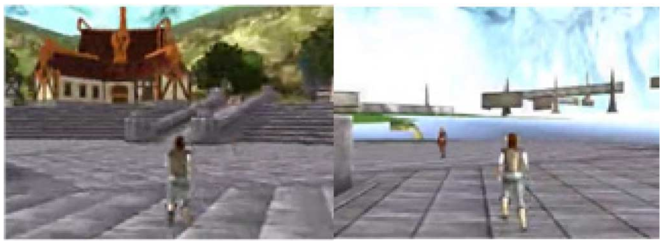


Fig. 11. Comparison of video frame quality for different rendering setting with 300 kb/s encoding bit rate. Left: view distance =150 m. Right: view distance = 70 m.

B. Overview of Content-Aware Adaptive Rendering Algorithm

Motivated by the tradeoff between rendering richness (inverse to $\mathrm{I_R}$) and video frame quality (inverse to $\mathrm{I_E}$), in this subsection we give an overview of the proposed CAAR algorithm, which aims at dynamically selecting the optimal rendering setting (a tuple of rendering factor values) in real time according to the network condition, such that CMR-UE is maximized at any time. However since the number of possible rendering settings may be very large, and the optimal decision depends on many factors such as application scenarios/network conditions, finding the optimal rendering setting will be complex and time-consuming. On the other hand, to be effective, rendering setting should be adapted in real time to cope with rapid network fluctuations. To resolve the above conflict, we propose to partition our approach into two parts: offline step and online step.

Fig. 12 shows the overview of the proposed CAAR approach, including the offline and online steps. The offline step derives an $\mathrm{I_E} - \mathrm{I_R}$ relation model for the tradeoff between rendering richness and video frame quality under different available video encoding bit rate budgets. The online steps are executed during CMR sessions; they utilize the $\mathrm{I_E} - \mathrm{I_R}$ relation model, together with the measured network condition information as well as the application content information (such as game scene information), to determine the optimal rendering setting in real time. As will be explained later in Sections V-D and V-E, the first online step uses the $\mathrm{I_E} - \mathrm{I_R}$ relation model to find the optimal $\mathrm{I_R}$ value that will maximize CMR-UE. Subsequently, the second online

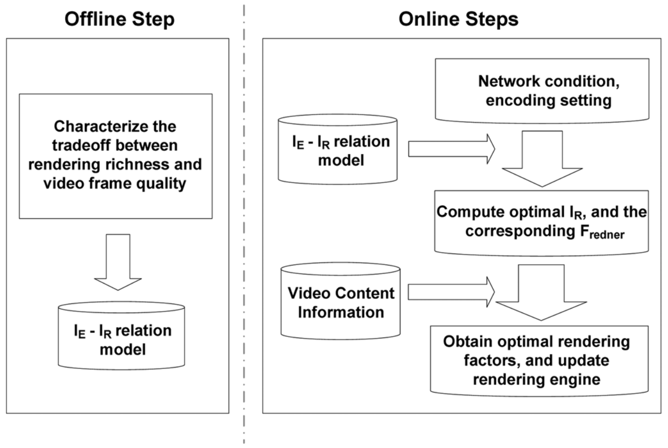

Offline Step
Online Steps
Characterize the tradeoff between rendering richness and video frame quality
I_E - I_R relation model
Network condition, encoding setting
I_E - I_R relation model
Compute optimal I_R, and the corresponding F_{redner}
Video Content Information
Obtain optimal rendering factors, and update rendering engine

Fig. 12. Overview of CAAR algorithm.

TABLE VIII
PARAMETERS FOR OFFLINE EXPERIMENTS

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

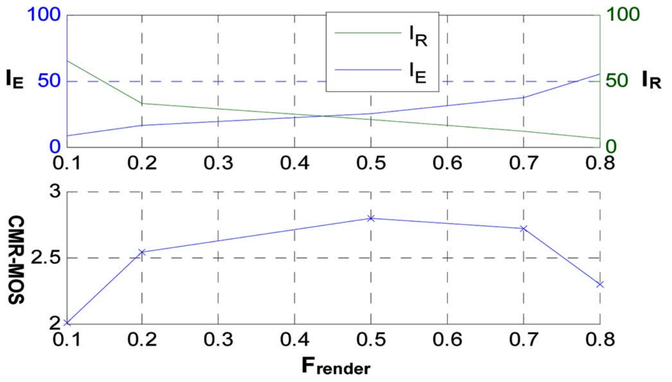


Fig. 13. I_E, I_R and CMR-MOS for different view distances used to render and encode a 200 kb/s CMR video.

step finds the optimal rendering setting which will lead to the optimal I_R value, using application content information.

C. *Offline Step: Automatically Characterizing the* $I_E - I_R$ *Relation Model*

In order to fully understand the tradeoff relation between rendering richness (inverse to the rendering impairment, I_R) and video frame quality (inverse to the encoding impairment, I_E), and their impact on CMR-UE, we need to conduct an offline experiment using the same framework shown in Fig. 4. During the entire offline experiments, we configure the network emulator such that the network bandwidth is sufficiently large and there is not any network impairment. Unlike the subjective user experiments described in Sections III and IV in which we need players to give their subjective evaluations, in this offline experiment, no human subject is needed as we utilize the CMR-UE model to compute the objective I_R and I_E values.

The procedure for the offline experiments is as follows. Note that while we describe the procedure using cloud gaming as an example, the same procedure can be followed for any CMR application. First, as the game is executed on the cloud server, we use a script to generate game control commands that automatically control the avatar to move around the virtual game world, including different kinds of game scenes. Second, while the rendered game video is captured and encoded, we vary the values of rendering and encoding factors, and record the corresponding PSNR values. For each possible combination of rendering and encoding setting, we execute the game for 10 s and record the average PSNR value during the 10 s. For each rendering setting i and rendering setting j, we compute the corresponding rendering impairment I_R using the CMR-UE model, and the encoding impairment I_E based on the average PSNR value, $PSNR^{ij}$, reported by the ×264 encoder. All the $I_E - I_R$ pairs are stored and tabulated in the CMR server to derive the $I_E - I_R$ relation model. Table VIII shows all the values of the rendering and encoding factors used for the offline experiments. Considering a total of 54 rendering settings and 24 encoding settings used, the total duration of offline experiments is: $54 * 24 * 10 = 12\,960$ s, which is less than 4 h of play time.

Fig. 13 shows the results of a representative offline experiment, when video bit rate is set to be 200 kb/s, video spatial resolution is set to be VGA, and rendering factor view distance is varied between 15 to 140 m, while keeping texture detail and realistic effect fixed at their highest values. As we fix the video encoding setting and change the view distance value, the corresponding I_R, I_E and CMR-MOS are shown in Fig. 13. Note that the offline experiment is conducted with different kinds of game scenes (indoor and outdoor), hence in the x-axis we show the corresponding F_{render} value instead of view distance. We can see from Fig. 13, as expected, when F_{render} increases, I_R reduces, but I_E increases. And because the decreasing slope of I_R is bigger than the increasing slope of I_E, the overall CMR-MOS increases initially with increasing view distance (and hence F_{render}). However, after the point when F_{render} equals 0.5, CMR-MOS starts to decrease as the decreasing slope of I_R is smaller than the increasing slope of I_E after that point. From the results shown in Fig. 13, we claim the optimal F_{render} for 200 kb/s bit rate target is 0.5, where it achieves the maximum CMR-MOS.

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

$$I_E = k^* I_R + b. \tag{6}$$

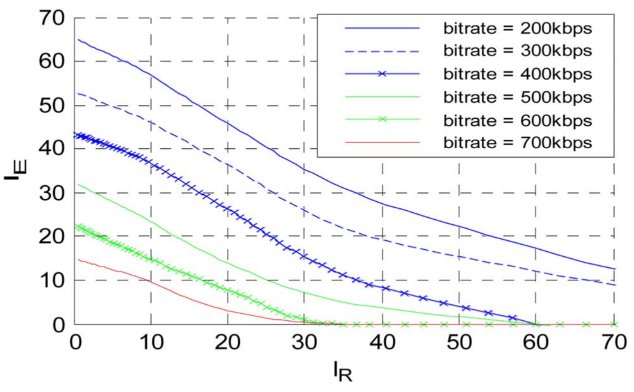


Fig. 14. Relation between $\mathrm{I_R}$ and $\mathrm{I_E}$ under different bit rate targets.

TABLE IX
COEFFICIENT VALUES FOR THE $\mathrm{I_E - I_R}$ MODEL (6) FOR GAME PLANESHIFT

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

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

Equation (6) and the coefficient values (like shown in Table IX) form the $\mathrm{I_E - I_R}$ relation model. Note that though the offline experiments and the regression analysis to develop this relation model should be done for each CMR application separately, the amount of required effort for each application is manageable because the procedure as described above is automated and typically takes a few hours of executing the application. In the next subsections, we will apply it to find the optimal rendering setting.

D. Online Step I: Determining Optimal I_R

After deriving the $\mathrm{I_E - I_R}$ relation model, as shown in (6), in this subsection and the next subsection, we will describe the online steps which aim at obtaining the optimal rendering setting (tuple of rendering factor values) under certain network constraints. The online step is divided into two steps, in step I, we find the optimal $\mathrm{I_R}$ values which will maximize the overall user experience CMR-MOS; in step II, we will translate this optimal $\mathrm{I_R}$ value into applicable rendering factor values.

We first formulate the problem of finding the optimal $\mathrm{I_R}$ value as the following optimization problem:

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

Find the optimal $\mathrm{I_R}$ value

$$\begin{aligned}\mathrm{I_R}^{\mathrm{opt}} &= \arg\max\ \mathrm{R}\\ &= \arg\max\{100 - \mathrm{I_E} - \mathrm{I_D} - \mathrm{I_L} - \mathrm{I_R}\\ &\quad + \sum_{\substack{i,j\in\{\mathrm{E,D,L,R}\}\\ i\neq j}} \mathrm{C_{ij}} * \sqrt{\mathrm{I_i} * \mathrm{I_j}}\}\end{aligned} \tag{7}$$

s.t.

$$\begin{aligned}\mathrm{I_E} &= \mathrm{k} * \mathrm{I_R} + \mathrm{b}\\ 0 &\le \mathrm{I_E}, \mathrm{I_R}, \mathrm{I_L}, \mathrm{I_D} \le 100.\end{aligned}$$

During a CMR session, the network packet loss L and the round-trip delay D are periodically measured by a network probing technique proposed in [14]; therefore the impairment due to packet loss, $\mathrm{I_L}$, and the impairment due to round-trip delay, $\mathrm{I_D}$, can be computed as described in our prior work [7], and can be treated as known values. Hence, the expression for R becomes

$$\mathrm{R} = \mathrm{M_1} - \mathrm{I_E} - \mathrm{I_R} + \mathrm{M_2} * \sqrt{\mathrm{I_E}} + \mathrm{M_3} * \sqrt{\mathrm{I_R}} + \mathrm{C_{ER}} * \sqrt{\mathrm{I_E} * \mathrm{I_R}} \tag{8}$$

where

$$\begin{cases}\mathrm{M_1} = 100 - \mathrm{I_L} - \mathrm{I_D} + \mathrm{C_{LD}}\sqrt{\mathrm{I_L} * \mathrm{I_D}}\\ \mathrm{M_2} = \mathrm{C_{ED}}\sqrt{\mathrm{I_D}} + \mathrm{C_{EL}}\sqrt{\mathrm{I_L}}\\ \mathrm{M_3} = \mathrm{C_{RD}}\sqrt{\mathrm{I_D}} + \mathrm{C_{RL}}\sqrt{\mathrm{I_L}}.\end{cases}$$

Furthermore, given the measured D and L values, a suitable video encoding bit rate is selected such that network delay D is reduced and response time constraints are satisfied, using the technique proposed in our previous work [6]. Knowing the video encoding bit rate, we can use Table IX to find the corresponding k and b values in the $\mathrm{I_E - I_R}$ relation model [(6)]. Substituting for $\mathrm{I_E}$ using (6), R in (8) can be expressed in terms of only one unknown factor, $\mathrm{I_R}$, and hence the optimization problem stated above becomes a 1-D optimization problem

$$\begin{aligned}\mathrm{R} = \mathrm{M_1} - (\mathrm{k} * \mathrm{I_R} + \mathrm{b}) - \mathrm{I_R} + \mathrm{M_2} * \sqrt{k * \mathrm{I_R} + \mathrm{b}}\\ + M_3 * \sqrt{\mathrm{I_R}} + \mathrm{C_{ER}} * \sqrt{(k * \mathrm{I_R} + \mathrm{b})\mathrm{I_R}}.\end{aligned} \tag{9}$$

To solve the 1-D optimization problem, one way would be to compute the derivative of function R [(9)], and find out at which point the derivative equals 0. However, the format of derivative of R is too complex for an analytical solution (closed-form solution). Therefore we turn to using computer simulation to find a numerical solution. We have done experiments to compute the derivative with MATLAB; it takes about 800 ms to solve the equation on a computer with Intel I5 processor, for granularity of e-5. However, the above computation time is not low enough to be used for the CAAR algorithm, considering the need to apply CAAR in real time, as well as the low response time needed for the CMR applications themselves.

Hence, we propose a time-efficient alternative approach of using recursive searching to find the optimal $\mathrm{I_R}$. The approach

```
Recursive Searching Approach
Input: packet loss impairment $I_L$, network delay impairment $I_D$,
coefficient k and b, and granularity threshold $\varepsilon$
Output: optimal $I_R$ value, i.e. $I_R^{opt}$

Initial search range  S = [$s_1$, $s_2$], $s_1$= 0, $s_2$ = 100;
Step 1: determine the number of iterations for granularity $\varepsilon$:
n =  Ceiling (log[($s_2$-$s_1$)/ $\varepsilon$])
Step 2: For k = 1 to n, repeat Step 3 and 4:
    Step 3: $p_1$ = ($s_1$+$s_2$)/2 - $\varepsilon$, $p_2$ = ($s_1$+$s_2$)/2 + $\varepsilon$;
    Step 4: If  R($I_R$= $p_1$) >= R($I_R$= $p_2$)
                Update the search range S, $s_2$ = $p_2$
            Else
                Update the search range S, $s_1$ = $p_1$
            Endif
Step 5: Return ($s_1$+$s_2$)/2
```

Fig. 15. Recursive searching approach to obtain optimal $\mathrm{I_R}$.

uses the following property of function R: R is a concave function of $\mathrm{I_R}$ and there exists one and only one maximum value. We have provided proof of the property online.[1] The recursive searching approach exploits this property of R, and instead of trying all possible values and comparing them, it perform a coarse-to-fine search and reduce the search space by half at each iteration, therefore achieve a low computation time.

Fig. 15 shows the procedure of the proposed recursive search algorithm. It exploits the concave property of R function and it is similar to the classic Dichotomous search algorithm [23]. We want to search for the optimal $\mathrm{I_R}$ value from a certain search range S, denoted as $[\mathrm{s_1}, \mathrm{s_2}]$. The function of R [(9)] is evaluated at two points close to the middle point of the search range: $\mathrm{p_1} = (\mathrm{s_1} + \mathrm{s_2})/2 - \boldsymbol{\varepsilon}$ and $\mathrm{p_2} = (\mathrm{s_1} + \mathrm{s_2})/2 + \boldsymbol{\varepsilon}$ where $\boldsymbol{\varepsilon}$ is a predefined small positive number. Then depending on whether $\mathrm{R(p_1)} > \mathrm{R(p_2)}$ or $\mathrm{R(p_1)} < \mathrm{R(p_2)}$, we decide that the optimal $\mathrm{I_R}$ lies in the left half or the right half of the search range, respectively. Then we reduce the search range by half and repeat this process. This process will end when the size of the search range is smaller than $\boldsymbol{\varepsilon}$, and the middle point of the last search range is returned as the optimal I_R. When we apply this recursive searching approach in MATLAB, the running time can be reduced to 68 ms for granularity of $\boldsymbol{\varepsilon} = \mathrm{e} - 5$.

E. Online Step II: Determine Optimal Rendering Setting

Next we describe how to find the rendering factors which ensure I_R^{opt} is achieved.

As stated in (1), the rendering impairment $\mathrm{I_R}$ equals to the sum of the impairments caused by the three rendering factors, with the impairment functions shown in (2), Table III and Table IV respectively. Although impairment functions for realistic effect and texture detail are in form of discrete tables (since there are limited number of options for these two rendering factors), the impairment function of view distance, $\mathrm{I_{VD}}$, is a continuous function which takes value between 0 and 100. This ensures that we can always find a combination of rendering factors to meet any I_R^{opt} target. For a specific I_R^{opt} target, we can select the realistic effect and texture detail first, and then subtract $\mathrm{I_{RE}}$ and $\mathrm{I_{TD}}$ from the I_R^{opt} to compute $\mathrm{I_{VD}}$, and finally determine the view distance.

For some I_R^{opt} values, we may have multiple choices for the three rendering factors. For example, if $I_R^{\mathrm{opt}} = 10$, either of the two options below can be used to achieve it: ($\mathrm{I_{VD}} = 10$, $\mathrm{I_{RE}} = 0, \mathrm{I_{TD}} = 0$), ($\mathrm{I_{VD}} = 6, \mathrm{I_{RE}} = 4, \mathrm{I_{TD}} = 0$). From the perspective of maximizing the CMR-UE, either of these options can be selected. In the following discussion, we will assume the realistic effect and texture detail are set to be their highest values ($\mathrm{I_{RE}}$ and $\mathrm{I_{TD}}$ are both 0). With this selection, the optimal $\mathrm{I_{VD}}$ will equal to I_R^{opt}, as shown in

$$I_{VD}^{\mathrm{opt}} = I_R^{\mathrm{opt}}. \tag{10}$$

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

The remaining problem is how to find the view distance, such that $\mathrm{I_{VD}}$ will equal to I_R^{opt}. Firstly, we need to compute the corresponding $F_{\mathrm{render}}^{\mathrm{opt}}$ using (2). Secondly, in order to achieve $F_{\mathrm{render}}^{\mathrm{opt}}$, we need to analyze game video content, and obtain the location information of all the 3-D objects and the camera. With these location information, we can obtain a distance set, $Dist = \{d_1, d_2, d_3, \ldots, d_n\}$. Each element of the set $Dist$, d_i, is the distance value from the ith 3-D object to the camera. We then sort these elements in ascending order and select the $\lfloor n * F_{\mathrm{render}}^{\mathrm{opt}} \rfloor$th smallest element as the optimal view distance, VD^{opt}, since this VD^{opt} value will ensure that the fraction of objects that are rendered is equal to $F_{\mathrm{render}}^{\mathrm{opt}}$, therefore will ensure the optimal $\mathrm{I_R}$ value, I_R^{opt}, is achieved.

Fig. 16 illustrates the flow of the online steps of CAAR algorithm, including online step I (highlighted using red box) and II. At short time intervals λ, the adaptation engine will check the network status (including D and L values), as well as video content information. On one hand, using the value of D and L, we can compute the optimal $\mathrm{I_R}$ value, I_R^{opt}, using the approach explained in online step I. After we get I_R^{opt}, we can compute I_{VD}^{opt} using (10) and the corresponding $F_{\mathrm{render}}^{\mathrm{opt}}$ using (2). On the other hand, using the 3-D objects and camera location information, we can obtain a distance set, $Dist$. We then sort these elements in ascending order and select the $\lfloor n * F_{\mathrm{render}}^{\mathrm{opt}} \rfloor$th smallest element as the optimal view distance, VD^{opt}. Finally, we will inform the graphic engine to update the view distance value to be VD^{opt}.

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

VI. Experimental Result in Real Wireless Network

In this section, we report on experiments conducted using a real commercial mobile network to verify the performance improvement possible by applying the proposed CAAR algorithm.

[1]Available online: http://esdat.ucsd.edu/JETCAS_paper.html

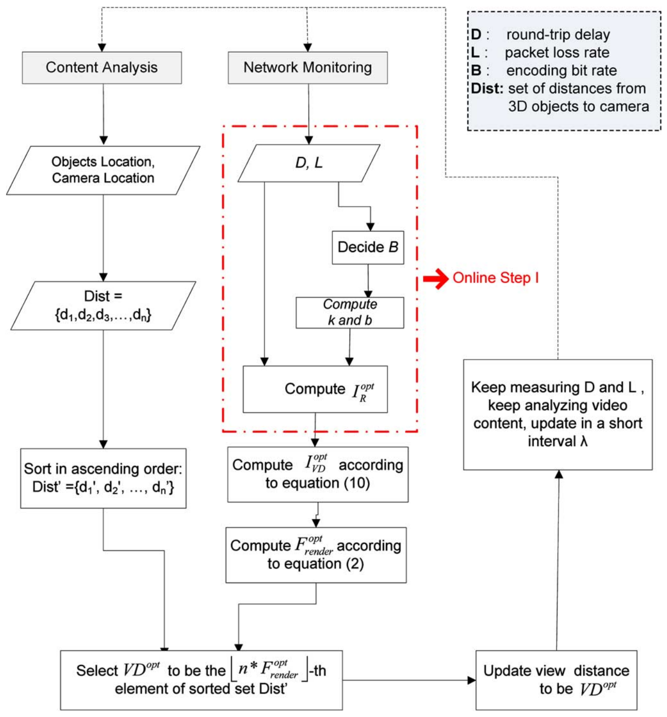

D : round-trip delay
L : packet loss rate
B : encoding bit rate
Dist: set of distances from 3D objects to camera
Content Analysis
Network Monitoring
Objects Location, Camera Location
D, L
Decide B
Online Step I
Compute k and b
Dist = {d1,d2,d3,…,dn}
Compute I_R^{opt}
Keep measuring D and L, keep analyzing video content, update in a short interval λ
Sort in ascending order: Dist' ={d1', d2', …, dn'}
Compute I_{VD}^{opt} according to equation (10)
Compute F_{render}^{opt} according to equation (2)
Select VD^{opt} to be the $\lfloor n * F_{render}^{opt} \rfloor$-th element of sorted set Dist'
Update view distance to be VD^{opt}

Fig. 16. Flowchart of the online steps for CAAR algorithm.

The experiments are conducted using two games, PlaneShift and Broadsides.

We have conducted extensive experiments to capture the bandwidth of a commercial 3.5G (1x-EVDO) network, using network measuring tool Iperf. We use a laptop as a mobile client, which is accessing the network through 3.5G network. We execute the Iperf software on the laptop, and hold it and roam around the UCSD campus at different times. The Iperf software will keep capturing the bandwidth trace which will be used later by the network emulator. Next, we use the experiment testbed shown in Fig. 4 to test the performance of CMG application in a 3.5G network. On the CMG server side, we use $\times$264 codec for video encoding, and use UDP as the transport protocol. Encoding adaptation technique [6] is applied to adapt the streaming bit rate to avoid network congestion. The mobile client is connected to the CMG server through the network emulator, which is running the captured bandwidth trace of 3.5G network.

For comparison purposes, we apply three different rendering strategies while running the same bandwidth trace on the network emulator: 1) *nonadaptive rendering*: adapt video encoding bit rate according to network condition, but fix view distance to be the largest value as decided by the rendering engine (hence no rendering impairment); 2) *noncontent-aware adaptive rendering*: adapt view distance purely based on encoding bit rate decided based on network condition, without considering video content information [6]; 3) *CAAR*: dynamically adapt the view distance using the CAAR algorithm discussed in Section V. Note that for each of the rendering strategies above, video encoding bit rate adaptation described in [6] is applied first depending on the measured delay and packet loss.

The *noncontent-aware* adaptation algorithm is based on our previous work [6]; it is different from the proposed CAAR algorithm in two aspects. 1) For each encoding setting, it selects the rendering setting that maximizes the video quality (PSNR), whereas our CAAR algorithm selects the rendering setting that maximizes CMR-MOS, which include both video quality and rendering richness. 2) It will select rendering setting only based on encoding setting, and not consider the differences in content among different scenes as done by CAAR.

Next, we present in details results of applying CAAR on the game PlaneShift. Fig. 17(a) presents a representative sample of the 3.5G mobile network bandwidth traces we collected and used in the experiments. It shows that the bandwidth of the mobile network can rapidly fluctuate. The duration of the network trace is 400 s, which we consider equally divided into four intervals A to D. We control the avatar to move in different game scenes during these four intervals: during interval A and C, the avatar is running in an outdoor game scene; during interval B and D, we control the avatar to move in an indoor game scene. By controlling the avatar in this manner, we can ensure that the game is played in indoor and outdoor scenes under both good and bad network conditions. Then in Fig. 17 from top to bottom, we sequentially present the results of encoding bit rate (after applying encoding adaptation technique), network round-trip delay, packet loss rate, view distance, rendering impairment I_R (inversely related to rendering richness), PSNR (indicating video frame quality), and resulting CMR-MOS score calculated by CMR-UE model.

Fig. 17(b) shows the encoding bit rates selected according to the available network bandwidth. The resulting round-trip network delay and packet loss rate are shown in Fig. 17(c). The results show that video encoding adaptation by itself can ensure the round trip delay is well maintained, most of the time to below 250 ms, only rising above 500 ms very occasionally. Same is true for the packet loss performance, with only a few spikes of high packet loss. However, as will be seen later, the video quality achieved (PSNR) and the overall user experience (CMR-MOS score) can be unacceptably low without applying rendering adaptation. In comparison, we will demonstrate that applying rendering adaptation, specifically content aware rendering adaptation, in conjunction with encoding bit rate adaptation, can significantly enhance not only the PSNR of the resulting encoded video, but also the overall user experience in spite of potentially compromising rendering richness.

Fig. 17(d) shows the view distance used by the graphics rendering engine to produce the rendered video frames. When no adaptive rendering is applied, the view distance is fixed at 150 m. When noncontent-aware adaptive rendering [6] is applied, the view distance is adapted according to the encoding bit rate shown in Fig. 17(b). When CAAR is applied, the view distance is adapted according to the encoding bit rate as well as the game content information: for interval B and D, as the avatar is in an indoor scene, the view distance selected and used is much smaller than that of interval A and C. The average PSNR values for the three strategies during the 400 s game session reported here are 30.28, 31.02, and 31.92 dB, using no rendering adaptation (only encoding bit rate adaptation), noncontent-aware rendering adaptation, and CAAR, respectively.

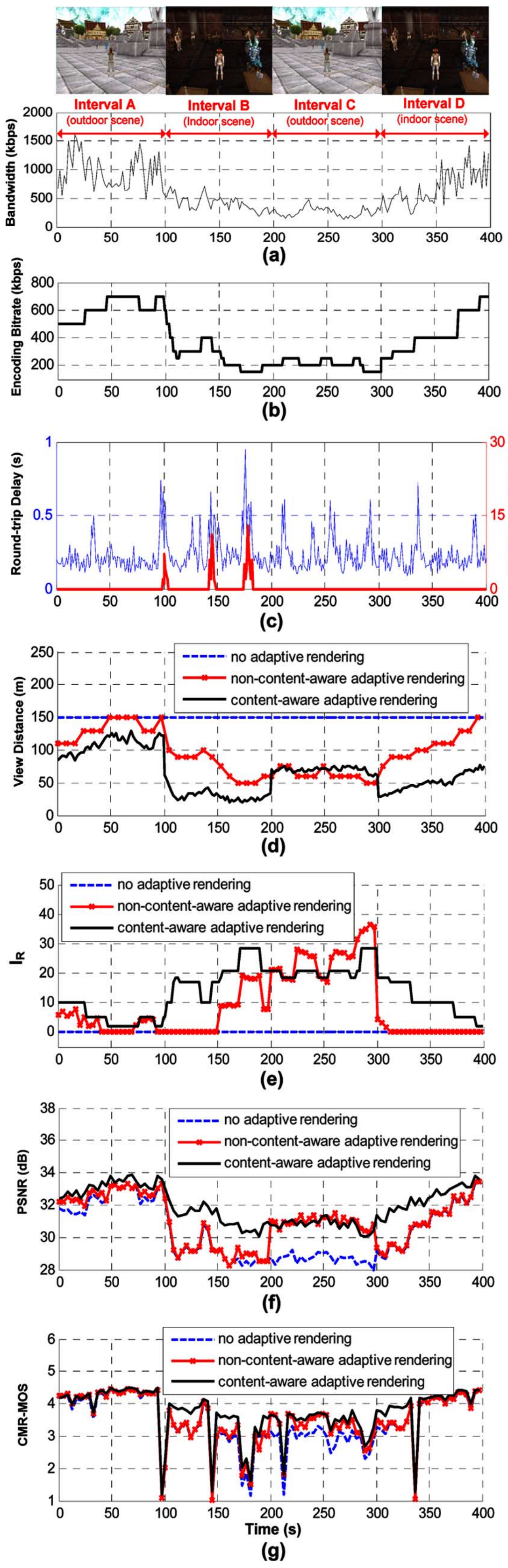


Fig. 17. User experience calculated using CMR-UE model on a representative 3.5G network for game PlaneShift.

We can see that CAAR algorithm can achieve a much better PSNR than the other two rendering strategies.

Fig. 17(e) shows the I_R values for different strategies. Higher I_R value indicates lower rendering richness. With no adaptive rendering, I_R is always 0; whereas for the other two adaptive rendering strategies, the rendering richness is adapted with the encoding bit rate. Fig. 17(f) shows the resulting PSNR performance. We can see that: 1) adapting encoding bit rate without adapting rendering (nonadaptive rendering) produces the worst PSNR performance since it does not reduce the content complexity when the encoding bit rate is low; 2) CAAR performs significantly better than noncontent-aware adaptive rendering by considering the content of the different scenes. For instance, let us compare the two adaptive rendering strategies during intervals B and C, in which the available network bandwidth are similar, but in which two different types of game scenes are rendered. CAAR assigns smaller view distance values during interval B (indoor scene) than during interval C (outdoor scene), thereby reducing content complexity of rendered video frames in interval B similar to frames rendered in interval C, hence enabling a satisfactory PSNR performance during both interval B and C. On the other hand, when noncontent-aware adaptive rendering is applied, it assigns similar view distance values for indoor and outdoor scenes, thereby rendering frames with larger content complexity for indoor scenes than outdoor scenes. Therefore while providing satisfactory PSNR performance during interval C, it produces low PSNR during interval B.

Fig. 17(g) demonstrates that the CAAR algorithm outperforms the other two strategies and yields the best CMR-UE performance during all the four intervals. This is because CAAR algorithm always selects the best tradeoff between rendering richness (inverse to I_R) and video frame quality (indicated by PSNR), and consider game content information to take care of all the scenes. The average CMR-MOS values for the three strategies during the 400 s game session reported here are 3.19, 3.34, and 3.55, using no rendering adaptation (only encoding rate adaptation), noncontent-aware rendering adaptation, and CAAR respectively.

Next, we briefly summarize results of conducting the same experiments as above using another 3-D game, Broadsides, which as described in Section IV is of a very different genre. Similar to the results for game PlaneShift (Fig. 17), the CAAR algorithm performs significantly better than the other two strategies. The average PSNR values for the three strategies during the 400 s game session are 30.02 dB, 31.17 dB, and 32.05 dB, using no rendering adaptation (only encoding bit rate adaptation), noncontent-aware rendering adaptation, and CAAR respectively. Similarly, the average CMR-MOS scores for the three strategies are 3.02, 3.28, and 3.50, respectively. Details can be found in . The above results demonstrate that the CAAR algorithm can achieve much better PSNR and CMR-MOS scores than the other two strategies, and is applicable and effective to other 3-D games of different genres.

VII. Conclusion

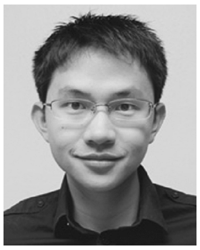
In this paper, we have studied a cloud based mobile video rendering and streaming approach which is promising to enable mobile users experience advanced rendering-based multimedia applications, such as 3-D gaming and augmented reality.

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

In this paper, we have demonstrated the use of adapting rendering richness and encoding quality to mitigate network conditions. However, too frequent or steep fluctuations in rendering richness or video quality may also impact user experience. In the future, we will extend the CMR-UE model to consider the impairment caused by the variations of rendering richness and video quality, and modify our CAAR algorithm accordingly.

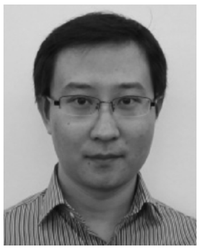
REFERENCES

[1] S. Richard and S. Gregor, "Peer-to-peer support for low-latency massively multiplayer online games in the cloud," presented at the NetGames Workshop, Paris, France, 2009.
[2] P. Ross, "Cloud computing's killer app: Gaming," *IEEE Spectrum*, vol. 46, no. 3, Mar. 2009.
[3] J. Ha and J. Jung, "Mobile augmented reality using scalable recognition and tracking," in *IEEE Virtual Reality Conf.*, Mar. 2011, pp. 211–212.
[4] B. R. Huang, C. H. Lin, and C. H Lee, "Mobile augmented reality based on cloud computing," presented at the Int. Conf. Anti-Conterfeiting, Securityand Identification, Taipei, Taiwan, Aug. 2012.
[5] S. Dey, "Cloud mobile media: Opportunities, challenges, and directions," in *Proc. IEEE Int. Conf. Comput., Network. Commun.*, Jan. 2012, pp. 929–933.
[6] 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.
[7] S. Wang and S. Dey, "Cloud mobile gaming: Modeling and measuring user experience in mobile wireless networks," *ACM SIGMOBILE Mobile Comput. Comm. Rev. (MC2R)*, vol. 16, no. 1, pp. 10–21, Jan. 2012.
[8] A. Khan and L. Sun, "Video quality prediction model for H.264 video over UMTS network and their application in mobile video streaming," in *Proc. IEEE Int. Conf. Commun.*, May 2010, pp. 1–5.
[9] V. Mukundan and C. Maninak, "Evaluate quality of experience for streaming video in real time," in *Proc. IEEE Global Commun. Conf.*, 2009, pp. 1–6.
[10] M. Ries, O. Nemethova, and M. Rupp, "Video quality estimation for mobile H.264/AVC video streaming," *J. Commun.*, vol. 3, pp. 41–50, 2008.
[11] K. Chen, T. Huang, P. Huang, and C. Lei, "Quantifying Skype user satisfaction," in *Proc. ACM SIGCOMM*, Oct. 2006, vol. 36.
[12] Y. Lu, Y. Zhao, F. A. Kuipers, and P. V. Mieghem, "Measurement study of multi-party video conferencing," in *Proc. Network.*, 2010, pp. 96–108.
[13] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "An evaluation of QoE in cloud gaming based on subjective tests," presented at the Workshop Future Internet Next Generation Netw., Seoul, Korea, Jun. 2011.
[14] S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *Proc. IEEE GLOBECOM'09*, Honolulu, HI, 2009, pp. 1–7.
[15] M. Bredel and M. Fidler, "A measurement study regarding quality of service and its impact on multiplayer online games," in *ACM SIGCOMM Workshop Netw. Syst. Support Games*, 2010.
[16] D. Prrritaz, C. Salzmann, and D. Gilllet, "Quality of experience for adaptation in augmented reality," in *Proc. 2009 IEEE Int. Conf. Syst., Man, Cybern.*, San Antonio, TX, Oct. 2009, pp. 888–893.
[17] W. Van *et al.*, "Scalable 3-D graphics processing in consumer terminals," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2002, vol. 1, pp. 369–372.
[18] N. Tack, "3-D graphic rendering time modeling and control for mobile terminals," in *Proc. Int. Conf. 3-D Web Technology*, 2004.
[19] PlaneShift [Online]. Available: http://www.planeshift.it/
[20] *The E-Model, a Computational Model for Use in Transmission Planning*, ITU-T Rec. G.107, Mar. 2005.
[21] 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*, Maui, HI, Jan. 2012, pp. 739–745.
[22] Broadsides [Online]. Available: http://cse125.ucsd.edu/cse125/2012/cse125g1/
[23] Dichotomous search algorithm [Online]. Available: http://shmathsoc.org.cn/lu/core%20part/Chap4.pdf



Yao Liu is currently working toward the Ph.D. degree at the University of California-San Diego, La Jolla, CA, USA.

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.



Shaoxuan Wang received the Ph.D. degree in computer engineering from University of California-San Diego, La Jolla, CA, USA.

He is currently a Scientist, Senior Staff Engineer in Broadcom Corp. He is the co-inventor of one U.S. and one international patents, with several others pending.

Sujit Dey (SM'03–F'13) 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), La Jolla, CA, USA, 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 also serves as the Faculty Director of the von Liebig Entrepreneurism Center. He is affiliated with the Qualcomm Institute, and the UCSD Center for Wireless Communications. 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 close to 200 publications, including journal and conference papers, and a book on low-power design. He is the co-inventor of 17 U.S. and two international 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.