Are All Games Equally Cloud-Gaming-Friendly? An Electromyographic Approach

Yeng-Ting Lee[12], Kuan-Ta Chen[2†], Han-I Su[2], and Chin-Laung Lei[1]
[1]Department of Electrical Engineering, National Taiwan University
[2]Institute of Information Science, Academia Sinica

***Abstract*—Cloud gaming now makes any computer game playable on a thin client without the previous worries and frustrations about the hardware requirements. It frees players from the need to frequently upgrade their computers as they can now play games that are hosted on remote servers with a broadband Internet connection and a thin client. However, cloud games are intrinsically more susceptible to latency than online games because game graphics are rendered on cloud servers and thin clients do not possess game state information that is required by delay compensation techniques.**

In this paper, we investigate how the response latency in cloud gaming would affect users' experience and how the impact of latency on players' experience varies among different games. We show that *not* all games are equally friendly to cloud gaming. That is, the same degree of latency may have very different impacts on a game's quality of experience depending on the game's real-time strictness. We thus develop a model that can predict a game's real-time strictness based on the rate of players' inputs and the game screen dynamics. The model can be used to simultaneously enhance players' gaming experience and optimize the operation cost of data centers.

I. Introduction

With the increasingly prevalence of cloud services, people are gradually moving their data to clouds and accessing their data remotely. In this way, less computation and storage resources are required on users' computers and therefore even mobile devices are capable of handling many computationally demanding tasks, rendering devices to the role of user interface to the supporting clouds. This approach has been successfully adopted by a variety of applications, notably web applications such as Google Docs and Dropbox, and video streaming by services like YouTube.

Computer gaming has also benefited from the support of clouds. Cloud gaming, also known as Games-on-Demand, is a technology which offloads the tasks of graphics rendering, in addition to computation and storage needs, into clouds. This technology makes any computer game playable on a thin client without previous worries of hardware requirements. As such, a great deal of interest has been generated for cloud gaming technology among entrepreneurs and the general public, and several startup companies offer or claim to offer cloud gaming services, such as OnLive[1], Gaikai[2], and OTOY[3].

*This work was supported in part by the National Science Council under the grant NSC100-2628-E-001-002-MY3.

†Corresponding author.

[1]`http://www.onlive.com/`

[2]`http://www.gaikai.com/`

[3]`http://www.otoy.com/`

A major attraction of cloud gaming is that it frees players from the need to frequently upgrade their computers as they can now play games that host on remote servers with a broadband Internet connection and a thin client. Here a thin client can be a lightweight PC, a TV with a set-top box, or even a mobile device. Consequently, there are no software installation overheads or compatibility issues if players wish to try a game because all the hardware and software is provided in the data centers by the game operators. Given these potential advantages for both game developers and consumers, cloud gaming has been considered a potential paradigm shift in the way computer games are delivered and played.

In the cloud gaming architecture, the game graphics are rendered in the cloud rather than by the client, which means that whenever a player makes a move, he cannot see the results until the cloud server processes the move, renders a new screen, and delivers the screen to the client. Therefore, *cloud gaming is significantly susceptible to the response latency*[4]. While the latency is also an issue for "traditional" network games, which render graphics on players' own computers, its impact on network games can be somewhat mitigated by using delay compensation techniques [1], such as dead reckoning [13]. However, such techniques cannot be applied to cloud games because their functions all require game state information, which are not available in cloud gaming clients. As a result, *cloud games are intrinsically more susceptible to latency than online games*.

In this paper, we investigate how response latency affects users' experience when playing games in clouds and how the impact of latency on players' experience varies across different games. We were initially motivated to pursue this investigation based on our observation that some games seem more playable than others in the cloud despite the inevitable latency, which is caused by network delay and processing delays that occur at both the server and client [4]. To pursue the answers to the above questions, we conducted user studies which involved nine games of three genres, namely, FPS (first-person shooter), RPG (role-playing games), ACT (action), and measure the players' emotions when different levels of latencies are introduced. Specifically, we quantify players' negative emotion against the latency based on the electrical

[4]For brevity, we may use "the latency" as a shorthand of the response latency which is the time required by a cloud gaming system to process a player's input and present the corresponding response to the player.

potentials produced by their corrugator supercilii muscle[5], the principle muscle which expresses frown and suffering. The results show that, in a cloud gaming setting, a game's pace and motion significantly affect how players perceive latency and that such phenomenon can be modeled on the rate of game inputs and the degree of game screen changes.

Our contribution in this paper is two-fold:

1) We verify that *not* all games are equally friendly to cloud gaming. The same degree of latency may have very different impact on a game's QoE (Quality of Experience) depending on the game's real-time strictness.
2) We develop a model that can predict a game's real-time strictness based on the rate of players' inputs and game screen dynamics. With the proposed model, one can easily assess a cloud game's sensibility to latency without conducting costly QoE measurement experiments [2]. The model can be used to enhance cloud gaming QoE in several ways. For example, our model can be used to detect game sessions which require prioritized processing to mitigate QoE degradation. Furthermore, it can help operators assign data centers to players to simultaneously optimize both player's gaming experience and data center operation cost (c.f. Section VI).

The remainder of this paper is organized as follows. The next section provides a literature review, and then we define necessary terms in Section III. In Section IV and Section V, we quantify and model a game's real-time strictness respectively. Finally we conclude in Section VII.

II. Related Work

Although Claypool [6] had already studied the suitability of games for cloud gaming in terms of motion and scene complexity, the main difference from our work is that, it focused on the bandwidth usage for real-time streaming of game screens; whereas our work focuses on the impact of latency. In [3], Chang et al. analyzed and modeled the QoE of thin-client gaming via general-purpose thin client, such as VNC, LogMeIn, and TeamViewer. Because such thin clients were not designed for gaming, their graphics streaming performance is not comparable with that of specialized cloud gaming systems.

In [10], Jarschel et al. conducted user studies to measure and model the QoE of OnLive during game play. They proposed a regression model to predict OnLive's QoE according to network performance metrics and players' experience and attitude towards the games. Our work complements [10] by answering the following unexplored questions: 1) Is the inevitable response latency due to cloud gaming architecture *game-dependent* in terms of user experience? 2) If so, how can a game's design affect its susceptibility to latency and how can we utilize such knowledge to improve the gaming experience and system resource allocation? We will investigate both issues in the rest of this paper.

[5]The corrugator supercilii muscle is located at the medial end of the eyebrow and beneath the forehead.

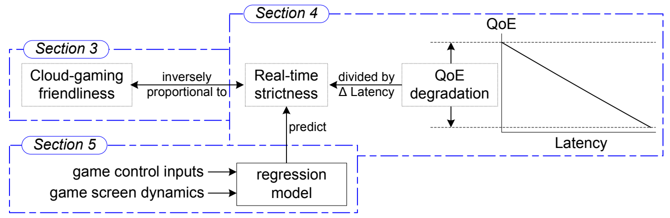

Fig. 1. An organizational overview of this paper.

III. Cloud-Gaming Friendliness

In this work, we define the "cloud-gaming friendliness" of a game as *a cloud game's susceptibility to latency in terms of its QoE*. That is, if a game can be played in a cloud without discomfort when moderate latency is present in the game's input-response loop, we consider the game to be *cloud-gaming-friendly*; otherwise, it is considered not cloud-gaming-friendly. Although a cloud game's QoE can be also affected by other restrictions, such as insufficient network bandwidth, we consider latency to be the most critical factor. The rationale for this is that, it is impossible to completely get rid of latency in cloud gaming systems because of the inevitability of game screen encoding and decoding time and network propagation delay, among other processing delays [4]. Therefore, we define a game's cloud-gaming friendliness as a function of how the game's QoE degrades in response to latency.

To quantify cloud-gaming friendliness, we define another notion called "real-time strictness". Real-time strictness is a measure of *a game's QoE deterioration when a certain amount of latency is introduced into the game's input-response loop*[6]. In other words, if a game's QoE is degraded by ΔQ when a certain degree of latency L is introduced, the game's real-time strictness (RS) is defined as $\Delta Q/L$. We can thus define a game's cloud-gaming friendliness to be RS^{-1}, as a game is more friendly (i.e., more playable) with the cloud gaming setting if its RS is lower (i.e., less susceptible to latency).

As can be seen in the structural overview of this paper illustrated in Figure 1, we will explain how we quantify the real-time strictness of a game in light of the empirical measurements in Section IV. Then, in Section V, we present a regression model that can predict a game's real-time strictness without conducting costly QoE measurements and then demonstrate the usefulness of the model in two applications.

IV. Quantifying The Real-time Strictness of Games

In this section, we present our methodology for quantifying the real-time strictness (RS) of games. We begin by describing the selected games, the QoE measurement approach, and experiment settings. We then present the measurement results and investigate the relationship between a game's genre and its real-time strictness.

A. Selected Games

In our study, we chose nine games of the three genres of ACT, FPS, and RPG. The three ACT games are LEGO Batman

[6]Here we make an implicit assumption that the magnitude of QoE degradation is approximately linear to the latency introduced. We will show the validity of this assumption via user studies in Section IV.

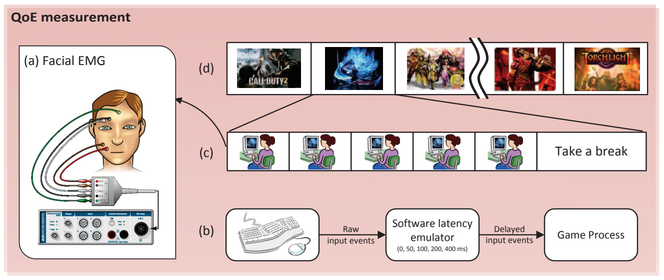

Fig. 2. (a) The setup for facial EMG recording with PowerLab. The black and gray electrodes are pasted on the location of corrugator supercilii (CS) muscle on a person's face. (b) The latency emulator injects delay into the message transmission path between the keyboard/mouse and game process. (c) and (d) describe the time line of our user study.

(Batman), Devil May Cry (DMC), and Sangoku Musou 5[7] (SM5). The three FPS games are Call of Duty: World at War (COD), F.E.A.R 2 (FEAR), and Unreal Tournament 3 (Unreal), while the three RPG games are Ys Origin (Ys), Loki: Heroes of Mythology (Loki), and Torchlight (Torch). These games were chosen with the goal to cover the spectrum of popular genres on the PC/console game market while obtaining three samples of each genre to understand the general characteristics of these genres.

B. QoE Measurement

Measuring a game's QoE, i.e., players' satisfaction, with the presence of various quality impairments has long been a challenging problem [5, 7]. Here we aim to measure how gaming experience degrades when a certain amount of latency is introduced, and for this reason, we need a game-independent QoE measure. We initially considered MOS (Mean Opinion Score) and SSCQE (Single Stimulus Continuous Quality Evaluation) approaches [12]. However, MOS is comparatively costly due to its fewer and coarser (i.e., five-level) responses. On the other hand, SSCQE requires subjects to continuously report their present gaming experience using a reporting tool such as a slider bar, which is not practical in our scenario because many games require both hands of players for the game controls. That is, SSCQE would likely disturb game play and diminish the flow experience [9].

We finally decided to use the facial electromyography (fEMG) approach [8] to measure players' experience for three reasons:

1) It provides *continuous* emotion measures (can be at a rate of 1000 Hz or even higher);
2) It *does not disturb* game play;
3) It is relatively objective since the emotional indicators are *directly measured* rather than told by subjects.

We use the fEMG potentials measured at the corrugator supercilii muscle, which is located at the medial end of the eyebrow and beneath the forehead, as indicated in Figure 2(a)[8]. As the fEMG potentials of the muscle are known to be associated with human's negative emotions [11], we believe that it is the perfect indicator of *how much a player is annoyed by the latency in a game's input-response loop*.

[7]Sangoku Musou 5 was also published under the title "Dynasty Warriors 6" in some regions of the world.

[8]Photo courtesy of ADInstruments.

TABLE I
SUBJECT AND TRACE SUMMARY OF OUR RS MEASUREMENT.

# Subjects	15 (14 males and 1 female)
Age	19–28 (mean 24.7 with std. dev. 2.9)
Game experience	Mean 4.6 years with std. dev. 1.3 years
Game play time each day	Mean 3.4 hours with std. dev. 1.1 hours
Total trace length	1,350 minutes
Avg. trace length each subject	90 minutes
Avg. trace length each game	150 minutes
# fEMG signals measured	$81,008,940$ samples
Avg. input rate	11.7 inputs/sec

C. Latency Emulation

In order to fully control the latency injected in the input-response loop of a game, we implement latency emulation on the subject's computer. To do so, we use `WH_KEYBOARD_LL` and `WH_MOUSE_LL` hooks (by calling the `SetWindowsHookEx` API) in Windows and delay all keyboard and mouse inputs made by subjects. Each of the input events is firstly queued in a buffer and later replayed after a desired time t, as illustrated in Figure 2(b).

To ensure that the emulated latency and the "real" latency occurring in cloud game play have the same impact on players' experience, we first conduct a separate study to compare their effects. In that study, we picked three out of the nine selected games (Section IV-A) that are available on the OnLive platform [14], namely, BioShock, F.E.A.R 2, and LEGO Batman. We then recruited 14 subjects to play the three games on OnLive. By measuring OnLive's game-specific processing delay [4] and injecting network delay using `wipfw`[9] when needed, we could control the overall input-response latency that players experience during OnLive game play. The games were played on a local computer with latency emulation, and played on OnLive with real and injected network delay combined, respectively. The Wilcoxon rank-sum test indicates that the fEMG potentials measured while subjects were playing games on both platforms are statistically equivalent with a significance level of 0.05 for all the latency settings 200, 300, 400, 500, and 600 ms[10]. Consequently, we proceeded with real-time strictness measurements using this latency emulation technique. This brought about two distinct advantages: 1) the full control of the latency injected into the input-response loop without restrictions from cloud gaming platforms; and 2) the free study of games that are not available on current cloud gaming platforms.

D. Experiment Setup

In our real-time strictness measurements, we emulated five latency levels, namely, 0, 50, 100, 200, and 400 ms in the input-response loop. In each round, we applied the five latency settings randomly with each setting lasting for 2 minutes, and each round thus spanning 10 minutes. Each subject was asked to play each of the nine games for a round and take a 5-minute

[9]`wipfw` is a packet filtering and accounting tool on Microsoft Windows. It can be used inject latency and packet loss and restrict bandwidth on specified connections.

[10]Note that the smallest latency setting is 200 ms because OnLive's processing delay is around 100 ms according to [4]; as such, we were unable to have an overall latency smaller than that.

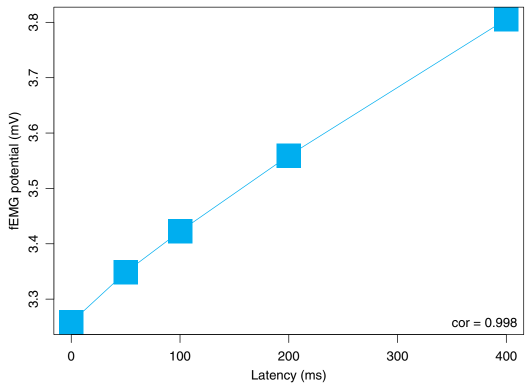

Fig. 3. The overall averaged relationship between the injected latency and the measured fEMG potentials in our study. The relationship is nearly linear with a Pearson correlation coefficient 0.998.

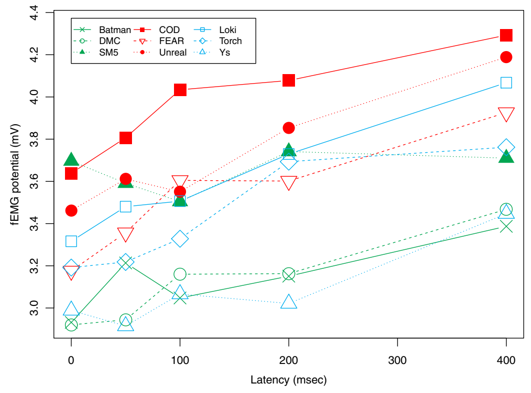

Fig. 4. The relationship between latency and average fEMG potentials.

break between successive rounds. In other words, each player took 130 minutes in our experiment, as indicated in Figure 2(c) and Figure 2(d). The order of the nine games presented was also randomized.

When measuring subjects' negative emotions during game play, the raw fEMG signals measured at the subjects' corrugator supercilii muscle were amplified and filtered by PowerLab 16/30 Bio Amp and recorded using LabChart. The signals were sampled at a rate of 1000 Hz with a band-pass filter of 10–200 Hz.

E. Results

We had 15 subjects (14 male and 1 female) participate in our study. Their ages ranged from 19 to 28 with an average of 24.7 years. A summary of the subjects' information is listed in Table I. To obtain an overview of how players' negative emotion was aroused by increased latency, we plot the averaged fEMG potentials over the 5 latency levels, as shown in Figure 3. The graph shows that, on average, *players' negative emotion was aroused approximately in proportional to the magnitude of latency*. The reason why fEMG potentials were not zero when there was no latency introduced is that since human's muscles have their own baseline activity, the fEMG potentials are always non-zero.

We then proceeded to inspect whether the arousal behavior of negative emotions was similar across different games. As is evident in Figure 4, all the games generally incurred higher

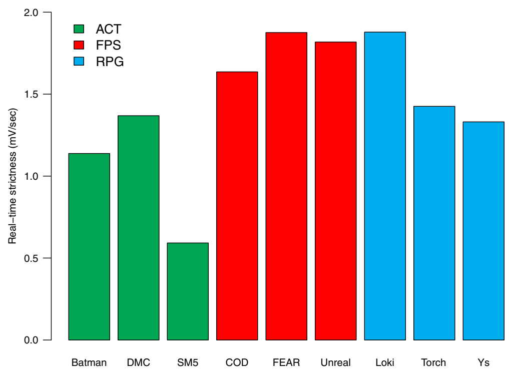

Fig. 5. The real-time strictness of the studied games.

fEMG potentials when the latency was longer. From the graph we can make two observations:

1) *The baseline fEMG potentials (i.e., those without latency injected) for each game are different.* This is because different games may cause unequal levels of tension depending on the game's presentation, sound, and plot, which may be an indicator of a game's capability to make players nervous and/or excited.
2) *The increasing rates of fEMG potential due to longer latency are game-dependent as well.*

In an effort to make the second point more clear, we plot the real-time strictness (RS) of each game as the range of a game's fEMG potentials divided by the range of latency difference (400 ms in our case), as shown in Figure 5. From the graph, we can see that generally the FPS games have the highest RS, followed by the RPG games, while the ACT games have the lowest RS. The fact that FPS games have a higher RS conforms to our intuition as this genre has long been considered the most latency-critical [7]. We consider the moderate to high RS of the RPG games reasonable because the three RPG games studied are all action-oriented RPGs, which are relatively faster-paced than regular turn-based RPGs. Moreover, we believe that the relatively lower RS of ACT games is due to the common "foolproof attacks" design of ACT games. That is, whenever the main character (i.e., the character controlled by the player) is near one or many opponents, the player just needs to repeatedly press the attack button/key without careful planning and timing control, and the main character will continually slash the opponents until they are all killed. This design is especially prominent in SM5, where repeated attack commands may invoke certain special attacks that last for 3–5 seconds and inflict more fatal damage. We believe that it is primary in virtue of this type of design that ACT games are less susceptible to latency and therefore have lower real-time strictness than FPS and RPG games.

V. Modeling The Real-time Strictness of Games

In this section, we investigate whether a game's real-time strictness (RS) can be modeled based on the game's design factors, such as game pace and screen update rate. If such a model can be obtained, we will no longer need to conduct costly QoE measurements, like the study we presented in

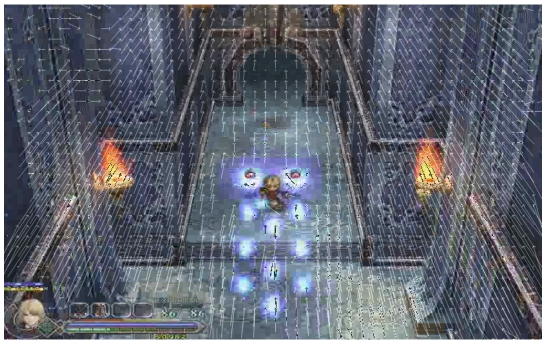

Fig. 6. A screen shot of Ys with its motion vectors overlaid. The motion vectors reveal how the macroblocks shifted since the last P- or I-frame in the recorded H.264 video. Because the avatar was moving downward, the motion vectors of the frame were generally pointing toward the top center on the screen.

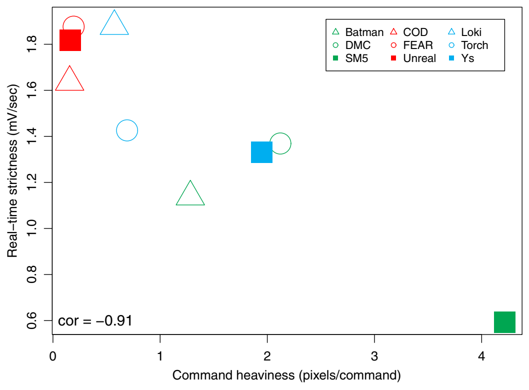

Fig. 7. The relationship between real-time strictness (RS) and command heaviness of the 9 selected games.

Section IV, in order to understand how a game's QoE degrades due to increased latency.

We conjecture that *how a game responds to players' commands is associated with its real-time strictness*. Our rationale is that most commands in games, such as move and attack, are associated with visual animations, which may not be instantaneous but span a few hundred milliseconds or even a few seconds. If a game's commands are mostly "heavy," i.e., associated with long and large amounts of screen changes, such as power attacks in SM5, the game will be less susceptible to latency because the timing of such attack commands is not so critical. On the other hand, if a game's commands are mostly "light," which correspond to simple, fast, and local moves, the game's real-time strictness will be higher. This means that players need to pay attention to each command and that the timing control of each command matters (in terms of the players' in-game performance).

We define the amount of screen changes in response to a player's command as *command heaviness*. And to verify our conjecture that the *command heaviness* is associated with real-time strictness, we resort to an empirical evidence. For this reason, in our user study (Section IV), we also recorded game screens as H.264 video and all the keyboard and mouse events issued by subjects during game play, which served as the basis for deriving the *command heaviness* factor. To do so, we extracted the motion vectors from the game play video recorded during the user study. Each frame of the video is broken down into macroblocks whose sizes are usually 4x4, 8x8, or 16x16 pixels. Given that a motion vector describes how a macroblock shifts its location in adjacent video frames, we can rely on it to capture how the game screen changes at a certain time. For example, in Figure 6, we plot a screenshot of Ys with its motion vectors overlaid. From the graph, we can see that the motion vectors are mostly pointing upward, as the main character was walking downward at the time the screenshot was taken.

Assuming that we have p subjects, each of which recorded a game play video containing f frames, and the jth frame of the video recorded by the ith subject have the motion vectors $\{mc_{ij1}, mc_{ij2}, \ldots, mc_{ijm}\}$, we compute the game's screen dynamics as

$$\sum_{i=1}^{p}\sum_{j=1}^{f}\frac{\mathrm{sd}(|mc_{ij1}|, |mc_{ij2}|, \ldots, |mc_{ijm}|)}{f\,p}/\text{inter-frame-time}, \tag{1}$$

where $\mathrm{sd}(\cdot)$ denotes the standard deviation function and $|mc|$ stands for the magnitude (i.e., length) of a motion vector mc. The final term inter-frame-time is used to normalize the screen dynamics factor so that the factor is comparable across games with different frame rates. We denote Equation 1 as *screen dynamics*, which has a unit of pixels/second.

To quantify a game's *command heaviness*, i.e., the amount of its visual feedback to each command, we normalize the *screen dynamics* (by Equation 1) by the game's input rate. The *input rate* is simply the average rate of commands (e.g., keyboard and mouse movement events) made by players during game play. Therefore, the *command heaviness* can be computed as

$$\text{command heaviness} = \frac{\text{screen dynamics}}{\text{input rate}}. \tag{2}$$

Since screen dynamics has the unit of pixels/sec and input rate has the unit of 1/sec, the unit of the resultant command heaviness is in pixels per command.

We plot the relationship between real-time strictness and command heaviness in Figure 7. As shown in the graph, the two factors have a strong, negative linear relationship with a correlation coefficient -0.91 and a p-value smaller than 10^{-4}. We find that the association of the two factors are sufficiently strong to build a regression-based predictive model:

$$\text{real-time strictness} = 1.8 - 0.28 \times \text{command heaviness} \tag{3}$$

with the adjusted R^2 equal to 0.82. We validate the predictive power of the model using the leave-one-out approach. That is, since we have derived the command heaviness factor for the nine games, we use eight out of the nine games as the "training dataset" to build a model according to Equation 3, and predict the RS of the remaining game (i.e., the "test dataset") using the model. This procedure is repeated nine times so that each game serves as the test dataset once. The validation results are plotted in Figure 8. As shown in the left graph of Figure 8, our model can estimate a game's RS even if the game is not included in model derivation. Generally speaking, the actual

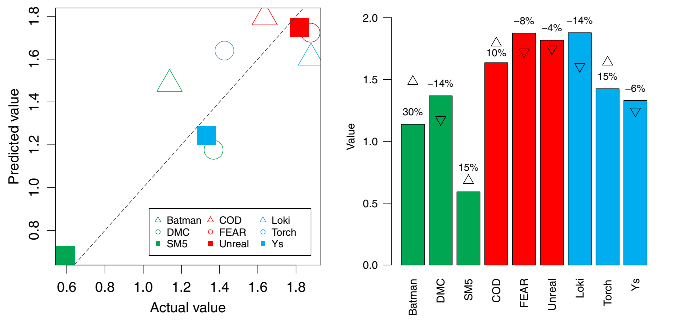

Fig. 8. (left) The scatter plot of the actual and predicted RS of the 9 games using the leave-one-out approach for cross validation. (right) The relative error of RS prediction in cross validation.

and predicted RS have a Pearson correlation coefficient of 0.87, while the average relative error between the actual and predicted RS is 12.88%. These figures clearly indicate that our model is not the result of model over-fitting. In sum, we consider that our proposed model, as shown in Equation 3, though quite simple, nonetheless possesses strong predictive power for games' real-time strictness.

VI. Applications

We believe that the proposed model can be helpful in a variety scenarios. For example, assuming that there are n online players, each of which is playing a game hosted on the same cloud game server, the n players have different network round-trip times to the server and the n game instances (with each instance corresponding to a player) possess different real-time strictness; as a consequence, the n players may perceive different levels of QoE degradation due to different response latencies (i.e., network latency plus latency due to cloud gaming systems). In this scenario, we can predict the degree of QoE degradation perceived by each player by multiplying each game's RS (predicted using Equation 3) and the player's response latency, which can be easily measured in run time. By so doing, we can infer which players are having a worse gaming experience than others, and accordingly prioritize the server's resources, such as CPU and GPU, to reduce those players' latencies and thereby mitigate QoE degradation they would otherwise experience.

As for another example, assuming that a cloud gaming operator hosts multiple data centers which are physically located at different places with varied operation costs. Whenever a user is signing in for game play in a cloud, the operator has two choices: 1) assign him a closer (from the user), but more expensive (in terms of operation cost) data center; or 2) assign him a more distant, but more inexpensive data center. The first option gives the player a shorter network delay but the operator needs to pay more for the operation, while the second option sacrifices the player's gaming experience in exchange for a lower operation cost. Which choice is better? The proposed model can be used to resolve this trade-off. With our model, we can compute how much QoE degradation the player will experience if either choice is adopted. As a result, we can estimate the total utility (by considering both operation cost and players' satisfaction) and assign to each player a data center which provides a "just good enough" gaming experience while saving as much operation cost as possible.

VII. Conclusion

In this paper, we have shown that not all games are equally friendly to cloud gaming. To achieve the goal of quantifying this friendliness, we have developed a model that can be used to predict a game's cloud-gaming friendliness based on how the game's screen changes spatially and dynamically and how frequently the game solicits players' inputs. In the future, we plan to incorporate more games in our study and further improve the power of the prediction model. In addition, we will investigate in more depth how the model can be used to simultaneously optimize cloud gaming server resource usages and players' gaming experience and help achieve a win-win situation for both the industry and gamers.

References

[1] Y. W. Bernier, "Latency compensating methods in client/server in-game protocol design and optimization," in *Proceedings of the 15th GDC*, Mar 2001.
[2] Y.-C. Chang, K.-T. Chen, C.-C. Wu, C.-J. Ho, and C.-L. Lei, "Online game QoE evaluation using paired comparisons," in *Proceedings of IEEE CQR 2010*, June 2010.
[3] Y.-C. Chang, P.-H. Tseng, K.-T. Chen, and C.-L. Lei, "Understanding the performance of thin-client gaming," in *Proceedings of IEEE CQR 2011*, May 2011.
[4] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proceedings of ACM Multimedia 2011*, Nov 2011.
[5] K.-T. Chen, P. Huang, and C.-L. Lei, "How sensitive are online gamers to network quality?" *Communications of the ACM*, vol. 49, no. 11, pp. 34–38, Nov 2006.
[6] M. Claypool, "Motion and scene complexity for streaming video games," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, ser. FDG '09. ACM, 2009, pp. 34–41.
[7] M. Claypool and K. Claypool, "Latency and player actions in online games," *Communications of the ACM*, Nov 2006.
[8] U. Dimberg, "Facial electromyography and emotional reactions," *Psychophysiology*, pp. 481–494, 1990.
[9] C.-L. Hsu and H.-P. Lu, "Why do people play online games? An extended TAM with social influences and flow experience," *Information and Management*, vol. 41, no. 7, pp. 853–868, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.im.2003.08.014
[10] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hossfeld, "An evaluation of QoE in cloud gaming based on subjective tests," in *Workshop on Future Internet and Next Generation Networks*, Jun 2011.
[11] J. T. Larsen, C. J. Norris, and J. T. Cacioppo, "Effects of positive and negative affect on electromyographic activity over zygomaticus major and corrugator supercilii," *Psychophysiology*, pp. 776–785, Sep 2003.
[12] N. Lodge and D. Wood, "New tools for evaluating the quality of digital television-results of the MOSAIC project," in *Broadcasting Convention, International (Conf. Publ. No. 428)*, Sep 1996, pp. 323–330.
[13] L. Pantel and L. C. Wolf, "On the suitability of dead reckoning schemes for games," in *Proceedings of ACM NetGames'02*, 2002.
[14] S. Perlman, "Introducing the OnLive game system," Nov 2010. [Online]. Available: http://blog.onlive.com/2010/11/17/introducing-the-onlive-game-system/