GECKO: Gamer Experience-Centric Bitrate Control Algorithm for Cloud Gaming

Yihao Ke[1,2], Guoqiao Ye[1,2], Di Wu[1,2(✉)], Yipeng Zhou[1,3], Edith Ngai[4], and Han Hu[5]

[1] Department of Computer Science, Sun Yat-sen University, Guangzhou, China
wudi27@mail.sysu.edu.cn
[2] Guangdong Province Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[3] Institute for Telecommunications Research, University of South Australia, Adelaide, Australia
[4] Department of Information Technology, Uppsala University, Uppsala, Sweden
[5] School of Computer Engineering, Nanyang Technological University, Singapore, Singapore

Abstract. Cloud gaming considered as the future of computer games enables users to play high-end games on resource-constrained heterogeneous devices. Games are rendered on remote clouds and delivered to users via the Internet in the form of video streaming, which can dramatically reduce the consumption of client-side resources. However, such service needs high bandwidth connections to make the game streaming smooth, which is already a major issue to hamper the prevalence of cloud gaming. In this paper, we propose a gamer experience-centric bitrate control algorithm called GECKO, to reduce the consumption of bandwidth resources for cloud gaming while only slightly impairing user quality-of-experience (QoE). Through measurement studies, we find that user QoE is mainly determined by the ROI (Region of Interest) size and QP offset. Hence, in order to save bandwidth consumption without severely impairing user QoE, we can lower the quality of the region outside of ROI. Our proposed GECKO algorithm is designed to adaptively tune the size of ROI and the quality of the outside region. We implement the GECKO algorithm on a real cloud gaming platform. The experiment results show that over 15.8% bandwidth can be saved compared with state-of-the-art approaches.

Keywords: Cloud gaming · Region of Interest · H.264 · Video coding Controller

1 Introduction

Cloud gaming as a fast-growing technology enables users to play high-end games with resource-constrained devices. Game scenes are rendered by remote clouds and delivered via the Internet in the form of video streaming. Technavio [1] predicted that the global cloud gaming market to grow exponentially at a compound annual growth rate (CAGR) of more than 29% during 2016–2020. However, high-speed bandwidth connections are required for smooth game streaming, which could severely restrict the prevalence of cloud gaming. Chen *et al.* [6] indicated that the bitrate of StreamMyGame varies between 9 Mbps and 18 Mbps, whereas Akamai [3] stated that the global average connection speed was 6.3 Mbps in the first quarter of 2016. In particular, the average speed of 65% countries is lower than 10 Mbps. The situation is worse for mobile users. There are only 18 countries serving mobile users with an average speed at or exceeding 10 Mbps.

Given the challenge of limited bandwidth, our work aims to propose a rate control algorithm on top of existing game streaming encoders to reduce bandwidth consumption without impairing user QoE. As stated in [11], major cloud gaming providers, e.g., Gaikai and Onlive, use H.264/AVC to encode gaming videos. To reduce bandwidth consumption, several rate control algorithms, e.g., CRF and ABR [4], have been implemented on H.264 by tuning parameters with the cost to harm user QoE. Different from the above works, our proposed GECKO algorithm tries to only lower the bitrate of the region that will not affect users' subjective feeling based on the fact that the human visual system (HVS) receives most of the visual information from the fixation points and around, regarded as Region-of-Interest (ROI) [15]. On the contrary, HVS's resolution falls rapidly from the point of gaze, hence we can propose a rate control algorithm to lower the quality of the region outside of ROI without harming user QoE much. Specifically, we first decide the size of ROI based on bandwidth conditions before we degrade the quality of the outside region. Intuitively speaking, the ROI size is enlarged if the bandwidth is not tight, otherwise it is shrunk.

Our contribution of this work is summarized as follows. We first conduct an in-depth measurement study to quantify the impacts of the ROI size and QP offset on the video bitrate and user QoE. The results indicate that it is feasible to lower down video bitrate with only slightly impairing user QoE by dynamically adjusting the ROI size and QP offset. We further propose a control-theoretic gamer experience-centric bitrate control algorithm for cloud gaming named GECKO, which can adaptively tune parameters according to video bitrate requirements. Finally, we implement our algorithm on top of x264, and use GamingAnywhere [8] for implementation and experiment to verify the effectiveness our algorithm.

2 Related Work

Different rate control or bit location schemes have been proposed in previous works. Sun and Wu [13] proposed a bit allocation scheme on macroblock layer based on Region of Interest (ROI). Their work assumed that the ROI of every frame is already known and then we can allocate different weights and bits to each macroblock. After getting the target bit of each MB, the QP of each MB can be computed by using the R-Q model proposed by JVT-G012.

Ahmadi et al. [5] introduced a conceptual Game Attention Model which determines the importance level of different regions of game frames according to user's attention. Subjective quality assessment showed that Game Attention Model helps to decrease the bitrate while maintaining the users' quality of experience (QoE). In cloud gaming, *Xue et al.* [18] found that delay and bandwidth played the important roles while improving the satisfaction of users. In order to improve the QoE of users, Some studies focused on which game server to connect [14], reducing the latency [22], improving the efficiency of transcoding [21] and streaming with minimal cost [17], or reducing the bandwidth consumption [7]. In this paper, we would explore control scheme associated with the bitrate of the ROI and reduce the bandwidth overload.

Shen et al. [12] proposed a novel rate control algorithm that takes into account visual attention. This work spent more efforts on ROI extraction and allocated bits for each macroblocks by local motion activity, edge strength and texture activity. *Yang et al.* [19] proposed a ROI-based rate control algorithm for video communication systems, in which the subjective quality of ROI can be adjusted according to users' requirements. In the proposed scheme, a Structural Similarity Index Map—quantization parameter (SSIM-QP) model is established.

3 Measurement Analysis of Bitrate Control for Game Streaming

To visualize how parameters affect streaming bitrate and user QoE and show the tradeoff between bitrate and QoE, we conduct a series of measurements in this section to quantitatively study the impacts of ROI size and QP offset.

3.1 Measurement Methodology

The game streaming is delivered to users in a sequence of frames, and each is composed by multiple macroblocks. The video quality (or bitrate) of the original game streaming is determined by quantization parameter (QP) of each macroblock. Larger QP value implies a larger step size in quantization and lower quality (or bitrate), and vice versa. Thus, by tuning the parameter QP, one can control the bitrate of the game streaming, which is a prevalent approach adopted by existing solutions. However, such trivial parameter tuning solution unavoidably harms user QoE.

In contrast, our control algorithm only adjusts QP for the region out of user interests to minimize the influence caused by lowering bitrate. We introduce one more parameter ROI size, which is defined as the ratio of the size of ROI compared with the size of the whole frame in the range from 0% to 100%. For simplicity, we assume that the ROI is a rectangular area located at the center of the frame. To control the bitrate, our algorithm will tune both ROI size and the QP for region outside ROI. In other words, only QP for the region out of ROI will be increased if it is necessary to reduce bitrate. For convenience, we define QP offset as the difference of QPs between macroblocks in and outside the ROI.

Note that QP offset and ROI size are two parameters irrelevant with the original streaming quality, and can be easily tuned by our proposed GECKO algorithm. For example, a QP offset of two means that the QP of each macroblock outside of ROI is increased by two compared with its original value; while the QP within ROI remains unchanged.

It is worth to mention that our approach is friendly for implementation since we only need to add a few auxiliary functions to existing streaming encoders, which will be introduced later.

In our measurements, we use the library x264 to generate the game streaming, which is a widely used software library for video streams encoding. The standard of the generated game streaming by x264 is H.264/AVC. According to the previous work [11], H.264/AVC is a *de facto* standard for cloud gaming.

The *pic_in* is one of the core data structure of x264 storing the properties and data of the input frame. In *pic_in→prop*, there is an array *quant_offsets* that controls the QP offsets for each macroblocks to be applied to this frame during encoding.

We control the streaming bitrate by altering the QP offsets stored in the array *quant_offsets*. Intuitively, QP offset value is set as zero for ROI and a positive value for region outside.

3.2 Metrics

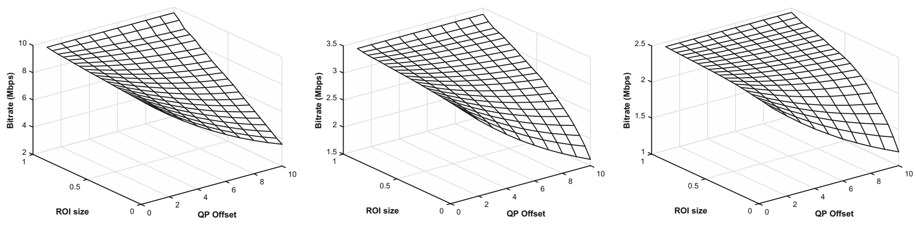
We use game streaming bitrate as the main metric to evaluate bandwidth consumption. We assume that more bandwidth is consumed if the game streaming with higher bitrate is delivered.

DSSIM (structural dissimilarity index) [10] is used as the metric to evaluate user QoE in this study. DSSIM can indicate how much distortion is incurred after we tune parameters with the rate control algorithm. DSSIM is defined as follows, $DSSIM = \frac{1}{SSIM} - 1$, where SSIM is the structural similarity index. SSIM is designed as the metric [16] to measure the similarity between two different images and DSSIM measures the dissimilarity instead. Thus, DSSIM is just computed by comparing the frame with altered parameters and the original frame. SSIM ranges from 0 to 1, thus DSSIM ranges from 0 to infinity. Higher DSSIM means lower QoE. The DSSIM depends on SSIM and it has been proved that SSIM is more consistent with human visual perception and significantly outperforms traditional measures, e.g., peak signal-to-noise ratio (PSNR) and mean squared error (MSE) [20].

There is a tradeoff between bandwidth consumption and user QoE. In spite that enlarging ROI size or decrease QP offsets for non-ROI will improve user QoE, the cost is more bandwidth consumption. In practice, due to limited bandwidth resources, we have to consider how to balance the tradeoff continuously.

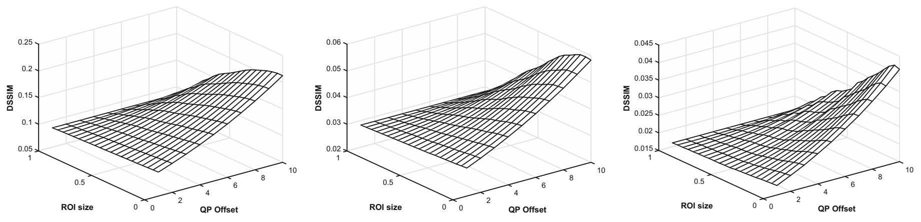
3.3 Insights from Measurement Analysis

In principles, bandwidth consumption and user QoE can be expressed as functions of the ROI size and QP offset, namely $bitrate = f(ROI\ size, QP\ offset)$



(a) Car racing (b) FPS (c) TBS

Fig. 1. Bitrate of each type of game.



(a) Car racing (b) FPS (c) TBS

Fig. 2. DSSIM of each type of game.

and $QoE = g(ROI\ size, QP\ offset)$. It is not difficult to accurately tune ROI size and QP offset, if we have exact expressions of functions f and g. However, unfortunately it is difficult to derive general expressions for f and g, which are affected by game type, gaming scene and so on. Specially, for game streaming with a sequence of frames, it becomes a tedious job because we have to create functions f and g for each frame in the worst case. Consequently, we turn to develop a control-theoretic gamer experience-centric bitrate control algorithm for cloud gaming to automatically tune parameters without the need explicitly derive f and g.

We select three most representative game types for numerical analysis by varying parameter values, including car racing game, first-person shooting (FPS) game and turn-based strategy (TBS) game. We generate a 30-s gaming video clip encoded by x264 encoder for each type of game. For each video clip, the ROI size varies from 0 to 100%, with a step of 5%; while the QP offset varies from 1 to 10, with a step of 1. We calculate the bitrate and DSSIM for each pair of parameter values.

Figure 1 shows how bitrate changes by varying ROI size and QP offset for each game video. x and y axes are parameter values, while z axis is the tuned video bitrate. In general, the trend is that the video bitrate will be lowered if the ROI size is shrunk or the QP offset is raised. By comparing different video types, we notice that the car racing game has the largest bitrate, due to the fast moving scenes with rich content. The TBS game has the lowest bitrate because the game scene changes slowly. The bitrate of FPS game is just in between that of car racing game and that of TBS game. When ROI size is set as 50% of the whole frame and QP offset is set as 5, the bitrates of these game videos will be reduced by 9.3%–21.9% compared with original videos. If ROI size is 0 and QP offset is 10, the bitrate will reduced by 51.1%–62.4%.

Figure 2 shows how DSSIM changes by varying ROI size and QP offset. The general trend is that DSSIM will be increased if we reduce ROI size or increase QP offset implying the worse user QoE. By comparing three videos, we find that car racing game has the largest DSSIM, which implies the worst QoE. The reason is that the fast moving scenes are very sensitive to parameter values. User QoE will drop sharply even if the bitrate is lowered a little bit. In contrast, the DSSIM of TBS game is the lowest implying the best user QoE because of slowly changing game scenes. The DSSIM will be increased by 25.1%–37% if the ROI size is 50% and QP offset is 5 compared to the original video. If we set ROI size to 0 and QP offset to 10, the DSSIM will be raised by 106%–156.5%.

Discussion: The tradeoff between bandwidth consumption and user QoE can be clearly observed by comparing Figs. 1 and 2. How to tune parameters to meet bandwidth constraints depends on the curve shapes plotted in Figs. 1 and 2, which are different for all three videos. This inspires us to propose a control-theoretic bitrate control algorithm in the next section.

4 GECKO—A Gamer Experience-Centric Bitrate Control Algorithm for Cloud Gaming

In this section, we turn to use control-theoretic algorithm that will automatically tune parameters by taking into account the feedback from the last time slot so that the long term average bandwidth constraints can be satisfied.

4.1 Problem Formulation

We assume that the Internet access services purchased by users will give them a certain bandwidth for streaming, which will not change in a short term. Then, our problem is how to control the video bitrate to meet the average downloading rate constraint as much as possible.

Define b^* as the target bitrate, i.e., the maximum tolerable bandwidth cost of cloud gaming. In this study, we can focus on the case of single gamer.

Consider a time-slotted system with time slot length of τ s. Define $D(k)$ and $R(k)$ as the QP offset and ROI size for time slot k respectively. Let b_k be the average bitrate in time slot k. We use DSSIM to represent the user QoE. Let q_k be the average DSSIM in time slot k.

By taking the tradeoff between the bandwidth (bitrate) cost and the QoE into account, we can define a generic utility function $\Phi(\cdot)$ to capture the impact of user preferences on the cloud gaming video bitrate. In this paper, we define the utility function $\Phi(\cdot)$ as a concave function of b_k. In general, the value of the utility function increases concavely with the increase of b_k. This property captures the fact that the marginal utility will decrease more significantly when b_k becomes larger.

4.2 Algorithm Design

Control theory [9] is an efficient methodology to solve our problem without the need to know the exact functions of f and g. The bitrate in the previous time slot can be used as a feedback signal for adjusting ROI size and QP offset. The value of b_k and q_k can be obtained at the end of each time slot k. Consequently, we can design an gamer experience-centric bitrate control algorithm for cloud gaming based on control theory [9] to optimize bandwidth cost. In this work, this proposed algorithm is named as **GECKO**. By controlling the update of ROI size and QP offset, we can approach the target control objective b^* gradually.

Although proportional-integral-derivative (PID) controller may have a better controlling performance, it relies heavily on the tuning of K_p, K_i and K_d, which are the coefficients for the proportional, integral, and derivative terms respectively. However, tuning these coefficients is not so easy. The coefficient of proportional controller has an intuitive interpretation—if the current bitrate is larger than the target bitrate, the coefficient of controlling ROI size should be a fraction to reduce the ROI size and vice versa. Thus, we adopt a proportional controller to solve the ROI size and QP offset allocation problem.

Let $\varDelta_b(k)$ be the marginal utility incurred by the difference between the current state and the target state, which is defined as below:

$$\varDelta_b(k) = \varPhi(b_k) - \varPhi(b^*). \tag{1}$$

We define two separate proportional control factors $G_R(k)$ and $G_D(k)$, where ψ_r and ψ_d are two positive constants which determine the smoothness of two control factors.

$$G_R(k) = \frac{1 + e^{\psi_r \cdot \varDelta_b(k)}}{2e^{\psi_r \cdot \varDelta_b(k)}} \quad , \quad G_D(k) = \frac{2e^{\psi_d \cdot \varDelta_b(k)}}{1 + e^{\psi_d \cdot \varDelta_b(k)}}. \tag{2}$$

If $b_k > b^*$, we should decrease the ROI size or increase the QP offset and vice versa. From the above definitions, the $\varDelta_b(k)$ is positive value at this time, and $e^{\psi_r \cdot \varDelta_b(k)}$ is a value larger than 1, consequently $G_R(k)$ is a fraction between 0 and 1, which satisfies our requirements. The $G_D(k)$ works in a similar manner. The properties of exponent function and the above control factor definitions also ensure the amplitude of adjustment will not be too large when b_k has a large difference with b^*, because the large amplitude of ROI size or QP offset adjustment may deteriorate user QoE.

The update of ROI size and QP offset can be governed by the following controllers:

$$R(k+1) = G_R(k) \cdot R(K) \quad , \quad D(k+1) = G_D(k) \cdot D(K). \tag{3}$$

The detailed description of our proposed algorithm is given in Algorithm 1.

5 Implementation and Performance Evaluation

In this section, we describe how to implement our proposed GECKO algorithm on x264 and GamingAnywhere [8] and conduct a set of experiments to evaluate the effectiveness of GECKO algorithm.

Algorithm 1. GECKO algorithm

```
Input:
    Target bitrate $b^*$;
    Time slot length $\tau$;
    User utility function $\Phi(\cdot)$;
1: Initialize: $k = 0$, $R(0)$ and $D(0)$ as the default values in cloud gaming server;
2: repeat
3:     Obtain the value of $b_k$ from the H.264 encoder.
4:     Use $b_k$ as the feedback signal to calculate control factors $G_R(k)$ and $G_D(k)$.
5:     Calculate ROI size and QP offset according to $R(k+1) = G_R(k) \cdot R(K)$ and
       $D(k+1) = G_D(k) \cdot D(K)$.
6:     Update ROI size and QP offset of the next time slot.
7:     Wait for a time slot $\tau$.
8:     Increase the slot index: $k = k + 1$.
9: until Quit playing cloud gaming.
```

5.1 Implementation on x264 and GamingAnywhere

The key point on the implementation of GECKO algorithm is how to obtain the bitrate and DSSIM for each frame. Obtaining the bitrate can be transformed into obtaining the frame size and then we can calculate the bitrate by the frame size and the number of frame-per-second (fps). Correspondingly, the return value of the core encoding function *x264_encoder_encode* is the encoded frame size. DSSIM is calculated by SSIM and we can obtain SSIM of each frame in x264. To enable SSIM computation, we set *param.analyse.b_ssim* $= 1$ when creating the encoder. After that, the SSIM of each frame is retrieved by *pic_out.prop.f_ssim* after encoding. After getting the bitrate and SSIM, we derive the average bitrate and DSSIM. Then we apply the GECKO algorithm to obtain the control factors $G_R(k)$ and $G_D(k)$ and update the $R(k+1)$ and $D(k+1)$. Finally, the new *quant_offset* array is produced and used in the encoding phase.

GamingAnywhere [8] is the first open source cloud gaming platform. From the version of *0.8.0*, it divides different functions into modules. Each module has several interfaces, such as *init*, *start* and *stop*. In this paper, we modified the *encoder_x264* module to implement our ROI rate control algorithm.

The modifications are listed as below:

- Add several variables in global states, including b^*, ψ_r, ψ_d, $G_R(k)$, $G_D(k)$, $R(k)$, $D(k)$ and *quant_offset* array.
- Add a controller function, which performs computation of $G_R(k)$ and $G_D(k)$ and update of $R(k)$, $D(k)$ and *quant_offset* array.
- Modify the *init* interface. In the initialization phase, we set $R(0)$ and $D(0)$ as the default values and retrieve the corresponding *quant_offset*. We also enable the flag of SSIM computation.
- Modify the *reconfigure* interface. The modified *reconfigure* interface calls the controller function to update the ROI size, QP offset and *quant_offset*.

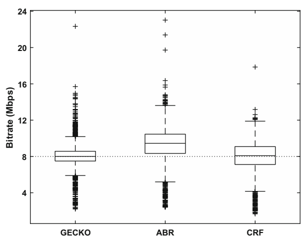


Fig. 3. Distribution of bitrate.

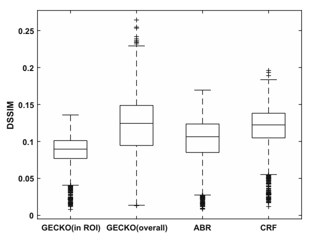


Fig. 4. Distribution of DSSIM.

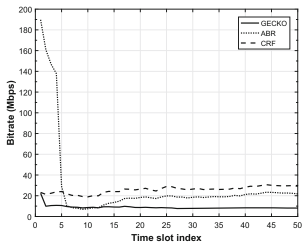


Fig. 5. Bitrate of first 50 time slots.

- Modify the *threadproc* interface, which calls the x264's encoding function. We assign *quant_offset* array to the *pic_in* struct before encoding and retrieve the frame size and SSIM.
- Add a *reconfigure* thread in *ga-server-periodic* core module which generates a *GA_IOCTL_RECONFIGURE* event every time slot and calls the *ioctl* interface of encoder module. The *ioctl* interface will later call the *reconfigure* interface.

5.2 Experimental Settings

We record a user's input of car racing game for 5 min. The input is then re-played into the GamingAnywhere to ensure the same input. The bitrate and DSSIM are retrieved during video encoding in GamingAnywhere. We adopt different rate control algorithms in the encoder module, including our proposed GECKO algorithm, ABR and CRF [4]. The ABR (Average Bit Rate) algorithm is a rate control algorithm targeting a specific bitrate on average. CRF (Constant Rate Factor) is the default rate control algorithm of x264, which aims to get the bitrate it needs to keep the requested quality level. The range of the factor is 0–51 and a lower value is a higher quality.

We define the utility function as a concave function like $\Phi(b_k) = \ln(b_k + 1)$. With the increase of b_k, the utility of a user will be increased but the marginal utility gain will decrease when b_k becomes larger. Note that other concave functions that have the similar properties can also be adopted. The time slot length τ is 0.1 s in following experiments. We set $b^* = 8$ Mbps. As for CRF algorithm, it is difficult to know the exact bitrate when we set the CRF factor. Thus, we experiment with several CRF factors and choose the best result whose bitrate is mostly closed to b^*. For GECKO algorithm, we set $\psi_r = 1$ and $\psi_d = 1$.

5.3 Experiment Results

In Fig. 3, it shows the ABR algorithm has the largest average bitrate. When setting the same bitrate target, GECKO algorithm can save about 15.8% of bandwidth compared with ABR. The result of CRF algorithm with a factor of 25 is the best one among all settings. The CRF algorithm and GECKO algorithm have better performance on controlling bitrate. And GECKO algorithm has a smaller deviation compared to CRF. What is more, it is not easy to adopt CRF algorithm in reality because the relationship between bitrate and CRF factor is not clear. Since CRF factor must be an integer, even tuning CRF factor with a smallest step (increase or decrease by one) will produce a relatively large bitrate variation.

Figure 4 shows the distribution of DSSIM of different rate control algorithms. Since the ABR algorithm produces a relatively large bitrate, it is not surprising to see its DSSIM is smaller than CRF and GECKO algorithms. The overall DSSIM of GECKO algorithm is worse than ABR and CRF, but the GECKO algorithm produces a lower DSSIM in the ROI, which confirms well with our assumption about lowering down video bitrate with only slightly impairing user QoE by dynamically adjusting the ROI size and QP offset.

Figure 5 shows the bitrate of each rate control algorithm in the first 50 time slots. The result indicates that our proposed GECKO algorithm converges to the target bitrate quickly while the other two algorithms converge slowly and have a large deviation from the target bitrate.

6 Conclusion

In this paper, we first conduct an in-depth measurement study to quantify the impacts of the ROI size and QP offset on the video bitrate and user QoE. The results indicate that it is feasible to lower down video bitrate with only slightly impairing user QoE by dynamically adjusting the ROI size and QP offset. We further propose a control-theoretic gamer experience-centric bitrate control algorithm for cloud gaming named GECKO, which can adaptively tune parameters according to video bitrate requirements. Finally, we implement our algorithm on GamingAnywhere and conduct a series of experiments to verify the effectiveness of our algorithm. The experiment results show that over 15.8% bandwidth can be saved compared with state-of-the-art approaches. In the future work, we will utilize low-cost gaze tracking devices, such as Intel RealSense [2] camera, to extract users' ROI and build a complete ROI-enabled cloud gaming platform.

Acknowledgement. This work was supported by the National Key Research and Development Program of China under Grant 2016YFB0201900, the National Science Foundation of China under Grant 61572538, the Fundamental Research Funds for the Central Universities under Grant 17LGJC23.

References

1. Global Cloud Gaming Market 2016–2020 (2016). http://www.technavio.com/report/global-gaming-cloud-market
2. Intel RealSense Technology (2016). http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html

3. Q1 2016 State of the Internet Report (2016). https://goo.gl/wS6wl2
4. Rate control definition in x264 (2016). https://goo.gl/rn80ct
5. Ahmadi, H., Zad Tootaghaj, S., Hashemi, M.R., Shirmohammadi, S.: A game attention model for efficient bit rate allocation in cloud gaming. Multimed. Syst. **20**(5), 485–501 (2014)
6. Chen, K.T., Chang, Y.C., Hsu, H.J., Chen, D.Y., Huang, C.Y., Hsu, C.H.: On the quality of service of cloud gaming systems. IEEE Trans. Multimed. **16**(2), 480–495 (2014)
7. He, J., Wu, D., Xie, X., Chen, M., Li, Y., Zhang, G.: Efficient upstream bandwidth multiplexing for cloud video recording services. IEEE Trans. Circuits Syst. Video Technol. **26**(10), 1893–1906 (2016)
8. Huang, C.Y., Hsu, C.H., Chang, Y.C., Chen, K.T.: Gaminganywhere: an open cloud gaming system. In: Proceedings of the 4th ACM Multimedia Systems Conference, MMSys 2013, pp. 36–47. ACM (2013)
9. Lee, E.B., Markus, L.: Foundations of optimal control theory. Technical report, DTIC Document (1967)
10. Loza, A., Mihaylova, L., Canagarajah, N., Bull, D.: Structural similarity-based object tracking in video sequences. In: 2006 9th International Conference on Information Fusion, pp. 1–6. IEEE (2006)
11. Shea, R., Liu, J., Ngai, E.C.H., Cui, Y.: Cloud gaming: architecture and performance. IEEE Netw. **27**(4), 16–21 (2013)
12. Shen, L., Liu, Z., Zhang, Z.: A novel H.264 rate control algorithm with consideration of visual attention. Multimed. Tools App. **63**(3), 709–727 (2013)
13. Sun, K., Wu, D.: Video rate control strategies for cloud gaming. J. Vis. Commun. Image Represent. **30**, 234–241 (2015)
14. Tian, H., Wu, D., He, J., Xu, Y., Chen, M.: On achieving cost-effective adaptive cloud gaming in geo-distributed data centers. IEEE Trans. Circuits Syst. Video Technol. **25**(12), 2064–2077 (2015)
15. Wandell, B.A.: Foundations of Vision. Sinauer Associates, Sunderland (1995)
16. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
17. Wu, D., Xue, Z., He, J.: iCloudAccess: cost-effective streaming of video games from the cloud with low latency. IEEE Trans. Circuits Syst. Video Technol. **24**(8), 1405–1416 (2014)
18. Xue, Z., Wu, D., He, J., Hei, X., Liu, Y.: Playing high-end video games in the cloud: a measurement study. IEEE Trans. Circuits Syst. Video Technol. **25**(12), 2013–2025 (2015)
19. Yang, L., Zhang, L., Ma, S., Zhao, D.: A ROI quality adjustable rate control scheme for low bitrate video coding. In: 2009 Picture Coding Symposium, pp. 1–4. May 2009
20. Zhang, L., Zhang, L., Mou, X., Zhang, D.: A comprehensive evaluation of full reference image quality assessment algorithms. In: 2012 19th IEEE International Conference on Image Processing, pp. 1477–1480. September 2012
21. Zheng, Y., Wu, D., Ke, Y., Yang, C., Chen, M., Zhang, G.: Online cloud transcoding and distribution for crowdsourced live game video streaming. IEEE Trans. Circuits Syst. Video Technol. (IEEE TCSVT) **27**(8), 1777–1789 (2017)
22. Zhou, L.: QoE-driven delay announcement for cloud mobile media. IEEE Trans. Circuits Syst. Video Technol. **27**(1), 84–94 (2017)