Enhancing Video Encoding for Cloud Gaming Using Rendering Information

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

***Abstract*— Cloud gaming allows games to be rendered on the cloud server and allows the rendered videos to be encoded and streamed in real time to the player's devices. Compared with other video streaming applications, cloud gaming offers a unique opportunity to enhance the video encoding process by exploiting rendering information. In this paper, we propose two techniques to improve cloud gaming video encoding, aiming at enhancing the perceived video quality and reducing the computational complexity, respectively. First, we develop a rendering-based prioritized encoding technique to improve the perceived game video quality according to network bandwidth constraints. We first propose a technique to generate a macroblock (MB)-level saliency map for every game video frame using rendering information. Furthermore, based on such a saliency map, a prioritized rate allocation scheme is proposed to dynamically adjust the value of quantization parameter of each MB. The experimental results indicate that the perceptual quality can be greatly improved using the proposed technique. We also develop a rendering-based encoding acceleration technique that utilizes rendering information to reduce the computational complexity of video encoding. This technique mainly consists of two parts. First, we propose a method to directly calculate the motion vectors (MVs) without employing the compute intensive motion search procedure. Second, based on the computed MVs, we propose a fast mode selection algorithm to reduce the number of candidate modes of each MB. The experimental results show that the proposed technique can achieve more than 42% saving in encoding time with very limited degradation in video quality.**

***Index Terms*— Cloud gaming, graphic rendering, macroblock (MB) mode selection, prioritized encoding.**

I. Introduction

DURING recent years, significant interest in cloud gaming has evolved, with commercial services like OnLive [1] and Gaikai [2] evoking interest among gamers, as well as attracting interest from researchers. Unlike traditional gaming, cloud games are rendered and executed on a cloud server, and the rendered game video is encoded and streamed to the thin client device through the network. The delivered game video will be decoded and displayed on the client device, while the game control signals are sent back to the cloud server for interaction.

Manuscript received September 22, 2014; revised February 7, 2015 and April 18, 2015; accepted June 12, 2015. Date of publication June 26, 2015; date of current version December 3, 2015. This work was supported by InterDigital, Inc., Wilmington, DE, USA. This paper was recommended by Associate Editor D. T. Ahmed.

The authors are with the Mobile Systems Design Laboratory, Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: yal019@ece.ucsd.edu; dey@ece.ucsd.edu; yaolu@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/TCSVT.2015.2450175

Although the cloud gaming approach is promising in enabling gamers to play rich 3-D games on thin client devices (such as mobile devices) without the need to download the entire game or purchase expensive graphics cards, there exist two challenges for this approach.

1) *Video Quality:* Since the encoded game video has to be streamed through the network that may have a limited and fluctuating bandwidth (especially mobile network), the video bit rate has to be adapted to cope with the available bandwidth. It is very challenging to maintain good video quality when network bandwidth is low. Moreover, unlike other video streaming applications such as video conferencing, cloud gaming has a high requirement for frame rate in order to ensure the fluidness of game. The high frame rate requirement makes it even more difficult to ensure a satisfactory video quality under limited bandwidth.
2) *Real-Time Encoding:* Cloud gaming is a highly interactive application, and the rendered video frames need to be encoded in real time. For example, consider playing a game with a frame rate of 30 frames/s, which is believed to be sufficiently high to provide a smooth playing experience. Then, in order to maintain a highly interactive playing experience, the encoding of each frame needs to be finished in 33.3 ms. The short encoding time budget is not easy to achieve if the game video is captured with large spatial resolution.

Unlike other video streaming applications such as Youtube or Skype, in cloud gaming, we have control over the game rendering process (the generation of video content) on the cloud server; hence, we can exploit some rendering information such as the camera location and angle, the location of each object, the distance of each object to the camera, etc., to help address the above two challenges.

In regard to the video quality challenge, we can leverage rendering information to exploit the perception redundancy in video encoding to provide a better perceptual video quality. Previous research [3] has shown that humans focus on small regions of the video rather than the entire video as a whole. Therefore, in this paper, we develop ways to use rendering information to identify the importance of different regions, spend more bits on the important regions, and consequently, achieve a better perceptual video quality compared with allocating bits equally on the entire video frame.

About the real-time encoding challenge, we propose a rendering-based video encoding acceleration technique to

exploit the rendering information to accelerate the encoding process. This technique consists of two parts.

1) We propose a method to directly calculate the motion vector (MV) between adjacent game frames and use this calculation method to replace the regular compute intensive search-based motion estimation method.
2) We propose a fast mode selection algorithm, which uses the homogeneity of MVs to reduce the number of candidate macroblock (MB) modes that need to be tested in the rate–distortion optimization (RDO) process, ultimately reducing the computational complexity of video encoding.

The rest of this paper is organized as follows. Section II summarizes the related work. Section III explains the rendering-based prioritized encoding technique, including the extraction of MB importance and the bit rate allocation algorithm, followed by the validation results. Meanwhile, in Section IV, we explain in detail how to use rendering information to help reduce the encoding complexity for cloud gaming, including how to use rendering information to compute MVs, and how to use rendering information to efficiently select the encoding mode for each MB. Finally, Section V concludes this paper and proposes some future work.

II. Related Work

In recent years, there has been a great deal of interest in cloud gaming [4]–[6], [14]. In cloud gaming, since the captured game video needs to be encoded in real time and streamed through networks, it is vital to maintain good perceptual video quality and achieve fast encoding speed in order to ensure a satisfactory interactive playing experience. There have been some studies on cloud gaming aiming at enhancing the video quality or encoding speed, but most of them are based on adapting/optimizing the rendering richness of the game scene. For example, Wang and Dey [8] analyzed the relation between video bit rate and rendering richness, and based on that, they proposed a rendering adaptation technique that adapts the rendering richness of game frame according to the network condition. Shirmohammadi [9] proposed to adapt the game scene complexity by omitting some less important objects (IOs) in the game frame when the network bandwidth is low. However, in this paper, we address the challenges of video quality and encoding speed from a different angle by: 1) optimizing the encoding process and 2) keeping the graphic rendering richness untouched.

In order to enhance perceptual video quality under a given network constraint (bit rate limit), prioritized encoding can be a promising solution. There have been various works on prioritized video encoding, which allocates more bits to regions of high importance in order to improve the video perceptual quality. However, for regular video, accurately extracting the importance information for different regions within a video frame is not a trivial task. In [10]–[13], techniques have been developed to extract the importance information using either spatial or temporal domain characteristics, such as luminance or chrome values. These techniques are very compute intensive and, therefore, not applicable for cloud game application that requires real-time encoding.

Prioritized encoding techniques specific for cloud gaming videos have not been very widely studied. Tizon *et al.* [15] have proposed a method to identify the importance of MBs based on the depth values (under the assumption that the closer objects have higher priority from viewers' perspective). However, this assumption is not always true since the most important game object may not be the closest object to the camera. In this paper, we propose a more general and flexible framework that allows object importance to be defined according to the depth as well as the game scene information, and therefore can correlate better with human perception and achieves better video quality.

Video encoding acceleration techniques specific for cloud gaming applications have not been widely investigated either. Taher *et al.* [16] proposed a high-level method to accelerate encoding by efficiently configuring the most important encoding parameters that can reduce the complexity while maintaining acceptable quality. However, instead of tuning the high-level parameters like in [16], our method takes a different approach, i.e., to use rendering information to optimize the motion estimation and mode selection process of H.264/Advanced Video Coding (AVC) [7] encoding.

On optimizing the motion estimation process, Fechteler and Eisert [17] have proposed a rendering-based method to calculate the MV using the motion information of every MB's central pixel. Similarly, Semsarzadeh *et al.* [18] have proposed to use game objects' motion information to bypass the regular motion estimation process. The two methods in [17] and [18] operate at MB level, and all the blocks within an MB have the same MVs, therefore may not achieve the best performance when encoding the MBs that contain multiple objects moving to different directions. In this paper, we propose to generate pixel-level MVs that can lead to higher motion estimation accuracy and encoding quality.

On optimizing the mode selection process, Cheung *et al.* [19] have also used depth information to accelerate the mode selection process. The method in [19] is based on the assumption that regions of similar depth are likely to correspond to regions of homogenous motion and, hence, are suitable for encoding using large partition. However, this assumption may not be true for MBs containing multiple small objects, which have a similar depth but move in different directions. The encoding acceleration technique proposed in this paper uses the motion homogeneity instead of depth, and therefore, it is not constrained by the above assumption.

Furthermore, Shi *et al.* [20] have proposed a 3-D image warping-assisted video coding method, which selects a set of key frames in a video sequence, uses the 3-D image warping algorithm to interpolate the other nonkey frames, and encodes the key frames and the residual frames separately. Although Shi's coding method achieves good encoding performance, there exist two limitations.

1) It works very differently from regular H264/AVC encoder, and hence cannot benefit from H264/AVC's features like motion estimation, mode selection, and variable block sizes.

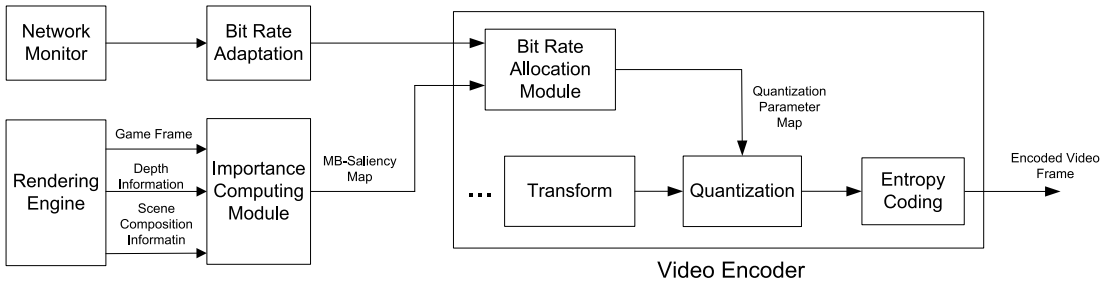


Fig. 1. Overview of rendering-based prioritized encoding technique.

2) A 3-D warping engine needs to be implemented both on the encoder and decoder, which increases the complexity.

On the other hand, our proposed technique follows the framework of regular H264/AVC encoder, and hence is much easier to implement and adopted by the cloud gaming providers.

III. Rendering-Based Prioritized Encoding Technique

In this section, we present a rendering-based prioritized encoding technique that aims at enhancing the video perceptual quality by adapting the encoding quality of the different regions of the video frame according to their importance. The proposed rendering-based prioritized encoding technique will first utilize rendering information such as pixel depth to compute the importance of different regions of game frame, convert this rendering information to an MB-level saliency map, and finally, find the optimal encoding parameter [in this paper, we choose quantization parameter (QP)] of each MB. The task of finding the optimal QP values for each MB is formed as an optimization problem. The optimal QP values are selected such that given the available bandwidth limit (bit rate budget), the perceptual video quality is maximized and the resulting video bit rate does not exceed the bandwidth limit.

This section is organized as follows. First, in Section III-A, we give a high-level overview of the rendering-based prioritized encoding technique including two key components: 1) the importance computing module and 2) the bit rate allocation module. In Section III-B, we first introduce how to extract the rendering information from the rendering engine. In Section III-C, we introduce how to use rendering information to calculate the importance of each MB. In Section III-D, we explain the approach to determine the optimal QP for each MB according to its importance and the total bit rate budget. Finally, in Section III-E, we discuss the experiment conducted to validate the performance of the proposed technique.

A. Overview of Rendering-Based Prioritized Encoding Technique

Fig. 1 shows an overview of the proposed rendering-based prioritized encoding technique. First, the rendering engine generates a game frame coupled with depth information (a pixel-wise map that indicates the depth of each pixel) and scene composition information (which pixel corresponds to which 3-D object and the importance of the object). Then, the rendering information will be fed to an importance computing module, which is responsible for combining the depth information and scene composition information together to generate an MB-based saliency map. Meanwhile, we use a network monitor proposed in [5] to periodically measure the interactive latency by sending probe packets and then adapting the encoding bit rate according to the measured latency based on the bit rate adaptation algorithm also proposed in [8]. This bit rate adaptation algorithm will dynamically select the appropriate encoding bit rate such that the video bit rate will not exceed the network bandwidth and cause network congestion. Moreover, the selected bit rate target and the generated MB-level saliency map will be used as inputs to a bit rate allocation algorithm, which is responsible for deciding the optimal QP of each MB, such that the overall perceptual video quality is maximized and the encoding bit rate target is met. Finally, the output of the bit rate allocation module, a set of the QP values for each MB, will be passed to the quantization module of the encoder and the game frame will be encoded using these QP values.

From implementation perspective, the proposed rendering-based prioritized encoding process will be executed in a frame-by-frame manner. The bit rate adaptation algorithm will be periodically executed with a period of 1 s, which allows enough granularity for bit rate switching to cope with network bandwidth fluctuations.

B. Extraction of Rendering Information

In order to develop a prioritized encoding technique that utilizes rendering information to determine the importance of different regions of a game frame, the first step is to clarify which rendering information we need, how they correlate with importance of regions in the rendered frames, and how we extract them. In this section, we will explain these three topics.

From a player's perspective, the *importance of a region* is defined by the probability/likelihood that region will attract the player's attention. In other words, regions with high importance are the regions that players will pay more attention to. In this paper, we suggest two kinds of rendering information that play significant roles in attracting human attention and can be conveniently extracted from the rendering engine: 1) depth information and 2) scene composition information.

First, in any rich 3-D game (especially role playing games), there is a virtual camera that generates the game video. One can easily make the observation that the objects closer to the camera (less depth) are more likely to attract player's attention [15]. Hence, the depth value would be a good

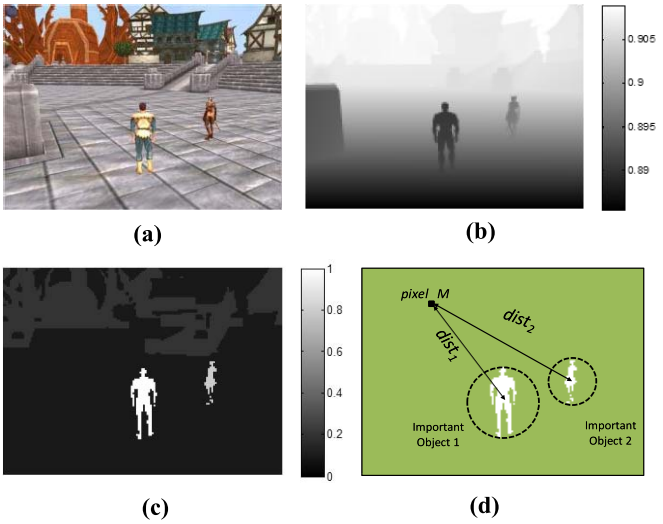


Fig. 2. Game frame and rendering information. (a) Top left: game frame of PlaneShift. (b) Top right: values stored in Z-buffer. (c) Bottom left: values stored in Stencil buffer. (d) Bottom right: illustration of distance-based saliency.

indicator for the importance of each pixel (smaller depth corresponds to higher importance). Moreover, the depth value of each pixel can be conveniently obtained using the well-known Z-buffer. During the rendering process, when 3-D objects are mapped to a 2-D plane, a depth value for each pixel is computed to determine object occlusion. After a game frame is rendered, the depth value of each pixel will be stored in the Z-buffer. One can easily read the depth values stored in the Z-buffer without interfering with the rendering pipeline.

Second, besides the depth value, another important factor that determines the amount of attention a player pays to a particular region is the scene composition information, i.e., which region contains which object and the priority of the object. For example, Fig. 2(a) shows a frame for the game PlaneShift [21], which is a typical 3-D role playing game. In this kind of game, the player controls the main avatar moving in a 3-D world and performing a set of tasks such as fighting against enemy avatars. In this game frame, the main avatar and the enemy avatar would undoubtedly be more important than the other objects such as the buildings and trees. Within the video frame, the regions containing the avatars will draw more attention from players.

The scene composition information is very helpful for identifying the importance of different regions. In general, this information is not available for regular video encoding and streaming applications, but cloud gaming offers the unique advantage for conveniently extracting scene composition information from the rendering engine. Similar to the depth values, we can obtain the scene composition information using the well-known Stencil buffer by following these two steps.

1) First, we need to define a priority value for objects in the game world. Take the game frame shown in Fig. 2(a) as an example; we can define the priority of the main avatar (the closer avatar) to be 1, the priority of the enemy avatar (the further avatar) to be 0.8, the priority of buildings to be 0.2, and other objects (like the sky and floors) to be 0.1.
2) After defining the priority value for each object, we can store this priority value into the Stencil buffer during rendering. Stencil buffer is a pixel-wise buffer supported by all the mainstream graphic libraries such as OpenGL and DirectX. Both of these graphic libraries have API functions, which allow game developers to control what data to write into the Stencil buffer during rendering. By configuring the Stencil buffer appropriately, whenever an object is being rendered, its corresponding pixels in the Stencil buffer will be written with its priority value. The values stored in the Stencil buffer can be easily read out to help determining the importance of different regions.

Note that although we have proposed prioritization of game objects, it is a very flexible step in our framework and it can be implemented in several ways.

1) *Prioritize All Objects:* If possible, the game developer can assign a saliency value to each object.
2) *Prioritize Some Objects:* The game developer can define some objects, which have certain functionalities or may affect the game logic, as IOs. Examples of these IOs are enemies, weapons, and doors. The game developers can assign only priority values to these objects and set the other objects' priority to be 0.
3) *Prioritize Only the Main Avatar:* The game developer can regard the main avatar as the only IO, set its priority to be 1, and set all the other objects' priority to be 0.
4) *No Object Prioritization:* In the case when game developers do not specify any object as IO, we will only use the depth information to determine saliency of a certain region.

Furthermore, the object prioritization step can be done offline and only once for a game. For a new game, it can be done during the game development period. Considering the amount of effort taken to develop a 3-D game, the additional effort for assigning object priorities should be negligible. Also note that the priority value associated with an object will be static and not dependent on game scenes.

Another scenario that should be taken into consideration is that when a game frame contains transparent/semitransparent objects. When a transparent/semitransparent object is rendered, for all the pixels covered by this object, their Stencil buffer values need to be updated to be the maximum value of this transparent/semitransparent object's priority and the old value stored in the Stencil buffer. For example, if in a game frame there is an enemy hiding behind a semitransparent glass, then the priority values stored in Stencil buffer will be the maximum value of the enemy object's priority and the glass object's priority.

C. Automatic Importance Computing Using Rendering Information

After explaining how to extract the rendering information (depth information and scene composition information), this section introduces the automatic importance computing module, which takes rendering information as input and computes the saliency for each MB as output.

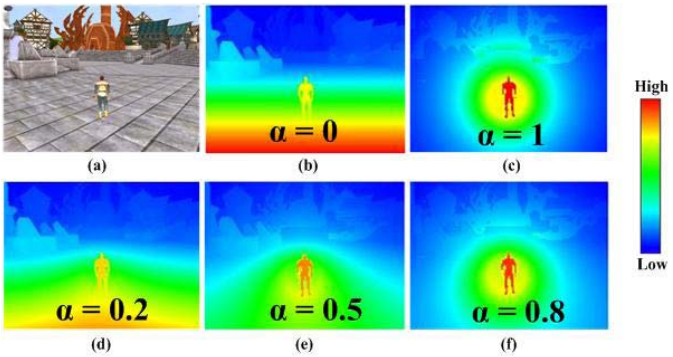


Fig. 3. Game frame and combined saliency map (S_{combine}) for different values of α.

As explained in Section III-B, we can configure the Z-buffer and Stencil buffer such that, after a game frame is rendered, the Z-buffer and Stencil buffer will contain the depth value and object priority value for each pixel. As an example, Fig. 2(b) and (c) shows the values stored in the Z-buffer and Stencil buffer, respectively, after the game frame shown in Fig. 2(a) is rendered.

The values stored in Z-buffer and Stencil buffer can be utilized to determine the importance of each pixel within the frame. In order to do that, we first define IO as the object whose priority value is larger than a certain threshold. As an example, in the game frame shown in Fig. 2(a), if we set a priority threshold of 0.6, then among all the objects in the game frame, only the two avatars have priorities higher than 0.6 [according to Fig. 2(c)]. Hence, there are only two IOs in the frame, as shown in Fig. 3(d).

We now propose two metrics to quantify the importance of each pixel using the depth information from the Z-buffer and the important region information derived using the Stencil buffer.

1) *Depth-Based Saliency (S_{depth}):* For each pixel, its depth-based saliency is a metric related to the depth of the pixel. The depth-based saliency for pixel M, $S_{\text{depth}}(M)$, can be computed using (1). In (1), $Z(M)$ is the value stored in the Z-buffer [a number between 0 and 1, as shown in Fig. 2(b)]. Larger S_{depth} value corresponds to lower depth and higher saliency

$$S_{\text{depth}}(M) = 1 - Z(M). \tag{1}$$

2) *Distance-Based Saliency ($S_{distance}$):* For each pixel, its distance-based saliency is a value related to its location and the location of all the IOs within the frame.

For a pixel M, in order to compute its S_{distance} value, $S_{\text{distance}}(M)$, two scenarios need to be considered.

1) If pixel M is inside an IO, then $S_{\text{distance}}(M)$ is equal to the priority of that IO.
2) If pixel M is not inside any IO, then $S_{\text{distance}}(M)$ is dependent on the distances from pixel M to the IOs of the game frame. Take Fig. 2(d) as an example. There are two IOs within that frame (the two avatars), and the distance-based saliency for pixel M should be computed as

$$S_{\text{distance}}(M) = \frac{1}{2 \times \log D}\left(P_1 \times \log \frac{D}{\text{dist}_1} + P_2 \times \log \frac{D}{\text{dist}_2}\right). \tag{2}$$

In (2), P_1 and P_2 are coefficients that represent the object priority of each avatar. Coefficients dist_1 and dist_2 are the distances from pixel M to the centers of the two IOs' minimum bounding circles (in unit of pixels), as shown in Fig. 2(d). D is the diagonal length of the video frame (in unit of pixels), and it can be regarded as constant in (2). Note that we use the diagonal length of video frame D to divide the distance values, dist1 and dist2, in order to get rid of the influence of video spatial resolution. We use the log function in (2) because it can distribute importance in a similar manner the human visual system works, such that the quality degradation starts to be perceived by the user only when the distances, dist1 and dist2, increase to a certain value. Ciubotaru *et al.* [23] have compared the performance of modeling this relation using a linear function and a logarithmic function. The experimental results show that the logarithmic function achieves a much better correlation with human perception than the linear function.

Equation (2) is for the game frames containing two IOs. In general, suppose a video frame contains T IOs, then $S_{\text{distance}}(M)$ is represented by

$$S_{\text{distance}}(M) = \begin{cases} P_i & (\text{if } M \in \text{IO}_i) \\ \dfrac{1}{T}\displaystyle\sum_{i=1}^{T} P_i \times \dfrac{\log \frac{D}{\text{dist}_i}}{\log D} & (\text{if } M \notin \text{IO}_i \quad \forall i \in [1, T]). \end{cases} \tag{3}$$

In (3), T is the number of IOs, P_i is the priority of the ith IO, and dist_i is the distance from pixel M to the center of ith IO's minimum bounding circle (only when pixel M is not inside of any IO).

After defining the *depth-based saliency* and *distance-based saliency* for each pixel, we combine them together to generate an MB-level saliency map. The procedure to compute the MB-level saliency map consists of the following steps.

1) Normalize the depth- and distance-based saliencies, respectively. Let $S_{\text{depth}}(M)$ be the depth-based saliency for pixel M. We first normalize $S_{\text{depth}}(M)$ as

$$S^N_{\text{depth}}(M) = \frac{S_{\text{depth}}(M)}{\overline{S_{\text{depth}}}} \tag{4}$$

where $S^N_{\text{depth}}(M)$ indicates the normalized depth-based saliency for pixel M and $\overline{S_{\text{depth}}}$ indicates the average depth-based saliency of all the pixels in the current frame. Similarly, we normalize $S_{\text{distance}}(M)$, the distance-based saliency for pixel M, as

$$S^N_{\text{distance}}(M) = \frac{S_{\text{distance}}(M)}{\overline{S_{\text{distance}}}}. \tag{5}$$

Note that during the normalization step (4), (5), to compute $S^N_{\text{distance}}(M)$ and $S^N_{\text{depth}}(M)$ for pixel M, we have already taken other pixels' distance-based saliency and depth-based saliency into consideration.

Hence, after the normalization step, the $S^N_{\text{distance}}(M)$ and $S^N_{\text{depth}}(M)$ values are good indicators for the relative importance of pixel M. Furthermore, this normalization step (4), (5) will execute frame by frame and it can automatically handle new emerging IOs. For the same pixel M, its $S^N_{\text{depth}}(M)$ and $S^N_{\text{distance}}(M)$ values will keep varying every frame, depending on how the current game frame is composed.

2) The normalized depth- and distance-based saliencies are next bounded as

$$S^N_{\text{depth}}(M) = \max\left\{S_L, \min\left\{S^N_{\text{depth}}(M), S_U\right\}\right\} \quad (6)$$

$$S^N_{\text{distance}}(M) = \max\left\{S_L, \min\left\{S^N_{\text{distance}}(M), S_U\right\}\right\} \quad (7)$$

where S_L and S_U are two constants to denote the lower and upper boundaries of $S^N_{\text{depth}}(M)$ and $S^N_{\text{distance}}(M)$. The thresholds S_L and S_U are used for calibrating saliency values in a similar manner the human visual system works, such that if the computed $S^N_{\text{depth}}(M)$ and $S^N_{\text{distance}}(M)$ values are too large or too small, we bound them to a reasonable range. The values of S_L and S_U are determined based on a study where we record multiple game video sequences, and then analyze the distribution of $S^N_{\text{depth}}(M)$ and $S^N_{\text{distance}}(M)$ for each game frame. We find that for all the frames, more than 90% of the pixels have their $S^N_{\text{depth}}(M)$ values distributed within [0, 4], and more than 90% of the pixels have their $S^N_{\text{distance}}(M)$ values within [0, 4]. Hence, we choose 0 and 4 as the default values for S_L and S_U, respectively. The details of the study can be found in [31].

3) Because the video is encoded at the MB level, we combine the pixel-level saliencies to compute the saliency at the MB level

$$S_{\text{combined}}(i) = \sum_{j=1}^{16}\sum_{k=1}^{16}\left[\partial \times S^N_{\text{distance}}(i,j,k) + (1-\partial) \times S^N_{\text{depth}}(i,j,k)\right] \quad (8)$$

where i is the MB index and j and k are the pixel indexes within one MB. As shown in (8), $S_{\text{combined}}(i)$ is a weighted average of $S^N_{\text{depth}}(i,j,k)$ and $S^N_{\text{distance}}(i,j,k)$. Coefficient α is used to control the relative weight between the depth-based saliency and distance-based saliency, and it takes a value between 0 and 1.

4) After Steps 1–3, the $S_{\text{combined}}(i)$ values of the spatial neighboring MBs within a local area would sometimes vary intensively. If the encoder allocates QP according to the $S_{\text{combined}}(i)$ values, the resulting visual quality of the neighboring MBs may vary significantly, leading to block artifacts. To solve this problem, a weighted 3×3 mean filter MF is applied to all the MBs in the game frame to smoothen the saliency map S_{combined}

$$\text{MF} = \begin{bmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{3} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \end{bmatrix}. \quad (9)$$

After Steps 1–4, we have converted the pixel-level saliency values (S_{distance} and S_{depth}) to an MB-level map S_{combined}. Fig. 3 shows an example of a game frame, and the associated S_{combined} with α varies between 0 and 1. From Fig. 3(b)–(f), we can clearly observe how the coefficient α can be used to control the relative weight between S_{distance} and S_{depth}. When α equals 0, we can tell from (8) that S_{combined} will be equal to S_{depth}, which is shown in Fig. 3(b); on the other hand, when α equals 1, S_{combined} will be simplified to S_{distance}, which is shown in Fig. 3(c). Fig. 3(d)–(f) can be regarded as a weighted average of Fig. 3(b) and (c), depending on the α value.

In order to determine the optimal α value for game PlaneShift, we have performed a study where we invite a group of subjects to play cloud game. As a subject is playing, we vary the α value from 0 to 1 and ask the subject to select the best α value. The best α value means the one that assigns the MB saliency closest to the subject's perception. According to the results, we find that most subjects choose 0.5 as the optimal value, and hence, in this paper, we heuristically select 0.5 as the default α value. The details of the experiment result can be found in [31]. Note that the optimal α value may be different for different kinds of games. If the cloud gaming service provider wants to experiment with different games, they can easily repeat this experiment and analyze the distribution of subjects' evaluations.

D. Rate Allocation Algorithm

In this section, we describe a bit rate allocation algorithm that aims at determining the optimal QP values of each MB, such that a specific bit rate budget is met and the overall visual quality is maximized.

First, for any MB MB_i, we adopt the linear relation model [24] between the compression distortion in terms of mean square error (MSE) and the quantization step q_i

$$\text{MSE}_i = k \times q_i + b. \quad (10)$$

Second, we adopt the rate model proposed in [25] that the encoding bit rate can be modeled using the quantization step q_i as

$$R_i = \frac{\theta}{q_i^{\gamma}}. \quad (11)$$

After introducing the relation between the distortion and quantization step (10) and the relation between bit rate and quantization step (11), we formulate the rate allocation task as the following optimization problem.

Given: Video bit rate target B_T.

Find the optimal quantization step for each MB q_i so as to

$$\min \text{WMSE} = \frac{1}{N}\sum_{i=1}^{N} S_{\text{combined}}(i) \times \text{MSE}_i \quad (12)$$

$$\text{s.t. } R = \frac{1}{N}\sum_{i=1}^{N} R_i \le B_T. \quad (13)$$

In this problem formulation, we use the weighted MSE (WMSE) as the object to minimize because WMSE is

the weighted average of the distortion for all the MBs, and it has incorporated the saliency value of each MB. Therefore, minimizing WMSE is equivalent of maximizing the human perceived video quality.

In the objective function WMSE, the saliency map of each MB, $S_{\text{combined}}(i)$, can be extracted from rendering engine and can be regarded as known parameters. Furthermore, according to (10) and (11), both MSE_i and R_i are the functions of the quantization step q_i. Hence, both the objective function and constraint function can be expressed as function of q_i, and we can use the well-known Lagrange method to solve this optimization problem.

First, according to the problem formulation, the corresponding Lagrange function can be stated as

$$f(q_1,\ldots,q_N)=\frac{1}{N}\sum_{i=1}^{N}S_{\text{combined}}(i)\times \text{MSE}_i +\lambda\left(\frac{1}{N}\sum_{i=1}^{N}R_i-B_T\right) \tag{14}$$

where λ is the Lagrange multiplier that needs to be determined. Substituting (10) and (11) into (14) yields

$$f(q_1,\ldots,q_N)=\frac{1}{N}\sum_{i=1}^{N}(k\times q_i+b)S_{\text{combined}}(i) +\lambda\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\theta}{q_i^{\gamma}}-B_T\right]. \tag{15}$$

To obtain the optimal solution for $\{q_i\}$, we use the Lagrange method

$$\frac{\partial f}{\partial q_1}=\frac{\partial f}{\partial q_2}=\frac{\partial f}{\partial q_3}=\cdots=\frac{\partial f}{\partial q_N}=\frac{\partial f}{\partial \lambda}=0. \tag{16}$$

By solving (16), we obtain the optimal q_i value

$$q_i^*=\left(\frac{\theta}{B_T N}\times\frac{\sum_{i=1}^{N}S_{\text{combined}}^{\frac{\gamma}{1+\gamma}}(i)}{S_{\text{combined}}^{\frac{\gamma}{1+\gamma}}(i)}\right)^{\frac{1}{\gamma}}. \tag{17}$$

We can observe from (17) that the optimal quantization step q_i^* is dependent on MB_i's importance, $S_{\text{combined}}(i)$. For a given MB, a high importance value will correspond to a low quantization step and good encoding quality. The q_i value is also dependent upon the model coefficients θ and γ [shown in (11)], which are determined by the video content characteristics and encoding parameters such as frame rate.

Finally, we convert the quantization step q_i to the QPs QP_i, since QP is the parameter that is adjustable in the encoder. The relation between QP and q_i specified in the H.264/AVC [7] standard is

$$\text{QP}_i^*=\text{round}\left[6\times\log\left(q_i^*\right)+4\right]. \tag{18}$$

The round function is used to ensure that the QP_i^* value is an integer such that the encoder can set it.

Fig. 4. Screenshot of game Broadsides.

TABLE I
PARAMETERS FOR VIDEO CODING

Codec	H.264/AVC
Total number of frames	1200 frames
Frame Rate	30 fps
Spatial Resolution	800x600 (PlaneShift), and 1280x720 (Broadsides)
GOP size	30 frames
Enable B-frames	No
Bit Rate Target	{600, 800, 1000, 1500, 2000} kbps

E. Subjective Assessment Experiments

In order to demonstrate the effectiveness of the proposed rendering-based prioritized encoding technique, in this section, we present the procedure and results of a subjective assessment experiment where we asked subjects to compare the quality of two sets of preencoded game videos. One set of test videos are encoded with regular H.264/AVC encoder, and the other set are encoded with the proposed prioritized encoding technique.

In order to demonstrate the generality and robustness of our proposed technique, our experiment—in addition to the previously introduced game PlaneShift—includes another game called Broadsides [22], which belongs to a different game genre. While PlaneShift is a role playing game, Broadsides is a first person shooter game. The screenshot of Broadsides is shown in Fig. 4.

We captured two raw game videos for PlaneShift and Broadsides, respectively. Then, we encode them using the typical parameter settings for H.264/AVC real-time encoding, as tabulated in Table I. As shown in Table I, each of these two raw videos has a length of 40 s. The spatial resolution is chosen to be 800×600 and 1280×720. In addition, the bit rate target varies between 600 and 2000 kbits/s. The model coefficient α [in (8)] is set to be 0.5. The coefficients θ and γ [in (11)] are set to be 7800 and 0.68.

For each raw video and bit rate target, two versions of encoded videos are created by encoding the same raw video twice. The first video is encoded in the usual manner such that the entire frame is encoded with the same quality. The second video is encoded using the proposed technique.

We recruited 22 subjects to participate in this experiment, most of whom are students from UCSD. The subject group included 14 males and 8 females with an age distribution ranging from 18 to 28. All of them have some prior experiences of playing video games.

The test procedure is conducted in a similar fashion to the recommendation from ITU-R BT.500-12, double-stimulus continuous quality scale [26] in order to evaluate the quality of the encoded video with and without the proposed

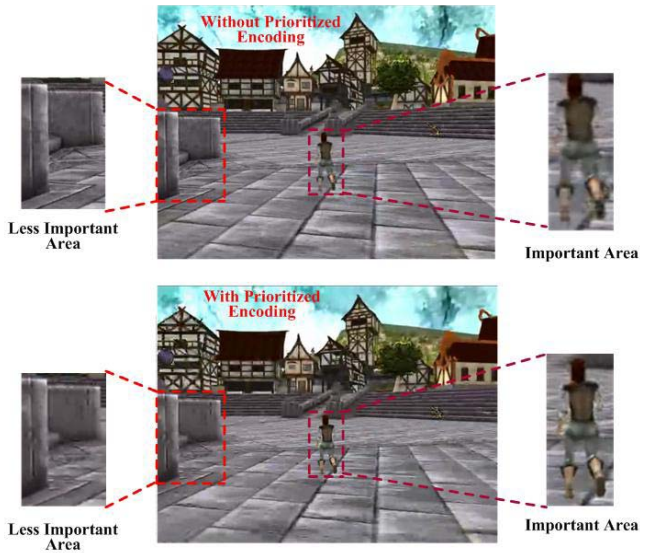


Fig. 5. Encoded video frames without and with rendering-based prioritized encoding, under the video encoding bit rate budget of 1000 kbits/s.

technique under a fixed bit rate target. The test was conducted indoor with good light condition. An iPad Air tablet with a customized application was used to play videos and collect participants' evaluations. The test procedure takes approximately 25 min, including 5 min of briefing and 20 min of user study. During the briefing, the subject was presented with a training video with instructions on how assessment will be conducted. The outlines of the 20-min user study are as follows.

1) For each game, five pairs of test videos are presented to the subject. Each pair corresponds to a bit rate target and consists of a normally encoded video and a prioritized encoded video.
2) The subject watches the test videos one pair at a time. Within each pair, the normally encoded video and the prioritized encoded video are presented in a random order, and the subject is unaware of the order.
3) After watching a pair of videos, the subject is asked to rate each of the two video's quality according to a five-point rating scale ranging from 1 (bad), 2 (poor), 3 (fair), 4 (good), and 5 (excellent). For each pair, the subject is able to switch between the two videos until he/she makes a decision.

As an example, Fig. 5 shows two encoded video frames without and with prioritized encoding, with the bit rate target set to be 1000 kbits/s. As we can see, for important areas such as the avatar, the video encoded without prioritized encoding has a much lower quality than the one encoded with prioritized encoding. Concurrently, for the less important areas of the game frame, rendering-based prioritized encoding leads to lower quality. Fig. 5 demonstrates the idea that the prioritized encoding increases the quality of important areas at the cost of quality degradation in less important areas.

The average subjective evaluations over all the subjects are shown in Fig. 6. We can see that for both games, the prioritized encoded videos lead to higher user experience than the normally encoded videos (without prioritized encoding).

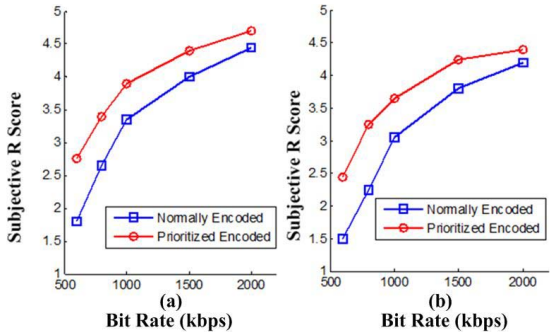


Fig. 6. Subjective evaluation of two games under different encoding bit rates. (a) Left: for game PlaneShift. (b) Right: for game Broadsides.

The gap between with and without prioritized encoding is big when bit rate target is low, and it will decrease as the bit rate target increases. This is due to the fact that, when the bit rate target increases to a certain value, the visual quality of the important areas is already sufficiently good and spending more bits on these areas will not further increase the visual quality, hence the advantage of prioritized encoding will not be as noticeable compared with the low bit rate cases.

Furthermore, in the experiment described above, except for video quality comparison, we have also compared the encoding time for the normal encoder and the proposed encoder. We find that the encoding times of the two encoders are very close, which means that the additional computational overhead introduced by our proposed technique is negligible. The details of the experimental results and explanation can be found in [31].

IV. Rendering-Based Encoding Acceleration Technique

The rendering-based prioritized encoding technique described in Section III aims at increasing the perceptual video quality under a fixed bit rate budget. We can think about this approach from another perspective that in order to achieve the same perceptual video quality, the prioritized encoding technique can help reduce the required video bit rate that needs to be streamed from the cloud servers. Hence, the proposed technique will be of interest for cloud gaming service providers since it can reduce the bandwidth consumption of the cloud server, and hence bring down the operational cost.

In this section, we propose another technique, referred to as rendering-based encoding acceleration (REA) technique, to further reduce the operational cost for cloud gaming by reducing the computational complexity of game video encoding.

It is well known that the most widely used video coding standard H.264/AVC achieves significantly higher encoding efficiency than the previous standards by adopting a set of new features such as adaptive motion compensation with variable block sizes, RDO technique, and so on. However, although H.264/AVC has a much higher encoding efficiency compared with the other standards, it also induces significantly higher computational complexity, which poses a challenge for applications like cloud gaming requiring real-time encoding and high interactivity.

The high computational complexity of H.264/AVC standard is mainly because of the search-based motion estimation and the large number of candidate modes. In order to address these two bottlenecks of encoding complexity, in this paper, we propose an encoding acceleration technique, which mainly consists of two parts.

1) We propose a direct MV calculation technique to utilize rendering information to directly calculate the MV for each pixel in a game frame, therefore bypassing the compute intensive search-based motion estimation process.
2) We propose a fast MB mode selection algorithm that exploits the motion homogeneity of the game frame and other rendering information, to efficiently determine a subset of candidate modes for each MB and skip the unlikely and unnecessary encoding modes.

This section is organized as follows. In Section IV-A, we described in detail how to use rendering information to directly calculate the MVs. In Section IV-B, we further reduce computational complexity by introducing a fast MB mode selection algorithm. Section IV-C presents the experimental results demonstrating the performance improvement achieved by the proposed technique.

A. Direct Calculation of Motion Vectors

In the H.264/AVC standard, temporal correlation in video sequence is exploited by block-based motion compensation, where the MBs of the current frame are predicted from the previously encoded frames. The removal of the correlation redundancy leads to high compression efficiency, but the search for the best MVs is a compute intensive process. Although researchers have developed various fast search methods for MVs, the search-based motion estimation still contributes to a big portion of the time taken by the video encoding process.

Fortunately, cloud gaming offers a unique opportunity to make the MVs much easier to be obtained. In cloud gaming, we have control over the rendering process (video content generation process), and we can use some rendering information to directly calculate the MVs with a set of calculations such as matrix multiplication, instead of using the compute intensive search-based approach specified in the H.264/AVC standard. In this section, we will explain this direct calculation method for obtaining the MVs.

The basic idea for direct calculation method is that, for each pixel in the current frame (assume that the corresponding screen coordinates is $\mathbf{S_c} = (sx_c, sy_c)$), we want to find the corresponding pixel location $\mathbf{S_p} = (sx_p, sy_p)$) in the previous frame (reference frame). Then, the MV for this pixel is the difference between these two locations ($\mathbf{S_c}$ and $\mathbf{S_p}$) multiplied by four to account for the quarter-pixel resolution of H.264/AVC [7].

In order to calculate the pixel location in the previous frame ($\mathbf{S_p}$), we need to obtain the following rendering information: the camera projection matrix for the current frame ($\mathbf{PM_c}$) and the previous frame ($\mathbf{PM_p}$) as well as the depth value for the pixel in the current frame (z_c), which can be conveniently extracted from the Z-buffer. The procedure to compute $\mathbf{S_p}$ is the following.

First, convert the pixel location (screen coordinates) in the current frame ($\mathbf{S_c} = (sx_c, sy_c)$) to the 3-D world coordinates ($\mathbf{W_c} = (wx_c, wy_c, wz_c)$) using the inverse of the projection matrix $\mathbf{PM_c^{-1}}$

$$\mathbf{W_c} = \mathbf{PM_c^{-1}} \times [sx_c, sy_c, z_c]^T. \tag{19}$$

Second, after obtaining the 3-D world coordinates in the current frame ($\mathbf{W_c}$), we need to find the corresponding 3-D world coordinate in the previous frame

$$W_p = \begin{cases} W_c & (\text{if } W_c \text{ belongs to a stationary object}) \\ W_c + (-V_o)T_f & (\text{if } W_c \text{ belongs to a moving object}). \end{cases} \tag{20}$$

As shown in (20), if the 3-D coordinates $\mathbf{W_c}$ belong to a static object, then its 3-D coordinates (coordinates in the game world) in the previous frame, $\mathbf{W_p}$, will be the same as the location in the current frame, $\mathbf{W_c}$. On the other hand, if $\mathbf{W_c}$ belongs to a moving object, such as a moving enemy avatar, we need to calculate $\mathbf{W_p}$ by considering the velocity of this moving object ($\boldsymbol{V_o}$), as well as the time difference between two consecutive frames ($\boldsymbol{T_f}$). The information of $\mathbf{V_o}$ and $\mathbf{T_f}$ can be conveniently obtained from the rendering engine.

Third, the 3-D world location $\mathbf{W_p}$ is reprojected to 2-D pixel location using the projection matrix of the previous frame

$$\mathbf{S_p} = \mathbf{PM_p} \times \mathbf{W_p}. \tag{21}$$

Finally, the MV is calculated as

$$\mathbf{MV} = 4(\mathbf{S_p} - \mathbf{S_c}). \tag{22}$$

Hence, for each pixel in the current game frame (at screen coordinate $\mathbf{S_c}$), we can compute its MV ($\mathbf{MV}$) through a few simple calculations such as matrix multiplication. This method is much faster than the regular search-based motion estimation process.

Although this direct calculation method is convenient, there exists two kinds of problematic cases where the proposed method cannot be applied.

1) New Emerging Objects: It is common in game videos that some objects will suddenly appear in a game frame, but did not exist in the previous frame. For the pixels and MBs that cover these new emerging objects, there exists no MV. We need to disable the proposed direct calculation method once we detect these pixels.

These pixels can be detected using the Stencil buffer. Each object in a game frame has a unique object ID. During the rendering process, as an object is being rendered, we can store the associated object ID in the correct locations of the Stencil buffer. As the game executes, whenever we detect a new object ID appearing in the stencil buffer that does not exist in the previous frame's Stencil buffer, we can mark this object as a new emerging object. For example, Fig. 7 shows an example of two consecutive frames of game Broadsides. Fig. 7(a) shows the reference frame (previous frame), and Fig. 7(b) shows the latter frame (current frame that needs to be encoded). The main difference between these two frames is that in the current frame, the ship has shoot some

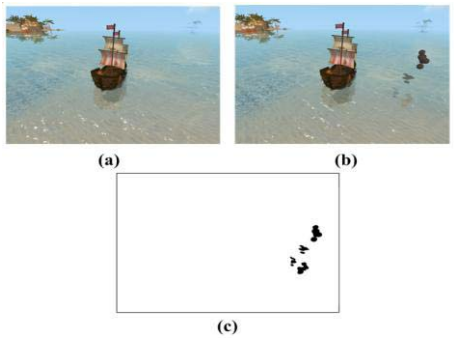


Fig. 7. Two consecutive frames of game Broadsides and the corresponding uncovered pixels. (a) Previous frame (reference frame). (b) Latter frame (current frame to be encoded). (c) Black pixels correspond to the *uncovered pixels* detected using the Stencil buffer.

black shells out, and these shells are the new emerging objects that do not exist in the reference frame. In Fig. 7(c), the detected new emerging objects using Stencil buffer are shown.

2) Pixels Pointing Out of the Frame: During the direct MV calculation process (19)–(22), it is possible that the compute pixel location in the previous frame (S_p) is not within the game frame's spatial range. An intuitive interpretation for this kind of situation is that the MV points out of the frame, and in this case, we need to disable the proposed direct calculation method as well.

In this paper, we will name the pixels that point out of the frame or cover new emerging objects as uncovered pixels. For these pixels, there exists no corresponding pixel in the reference frame and our proposed rendering based direct calculation method will not work. Furthermore, if an MB contains uncovered pixels, we name it uncovered MB. As will be explained later, in our proposed encoding acceleration technique, we will treat these uncovered MBs differently from other MBs. For these uncovered MBs, we need to use regular search-based motion estimation (ME) methods to obtain their MVs.

Note that the output of the direct calculation technique will be pixel-level MV. As will be explained in the next section, the pixel-level MV will first be exploited to determine the candidate mode set for each 16×16 MB. Then, in the following step of computing the rate-distortion (RD) cost for each possible mode, the pixel-level MV will be combined into block-level MV. In this paper, we propose to use the average vector of all the associated pixels' pixel-level MV as the block-level MB.

B. Fast Mode Selection Algorithm

As introduced in Section IV-A, H.264/AVC standard uses RDO technique to select the optimal mode from a set of candidate modes with block size ranging from 16×16 to 4×4.

In the scope of intra encoding, large size MB partition is suitable for smooth areas with low spatial complexity, and small partition is suitable for detailed areas with high spatial complexity. In the scope of inter modes where motion compensation is used to reduce temporal redundancy, large partition is suitable for the MBs exhibiting homogeneous motion properties, and small-size MB is suitable for MBs with

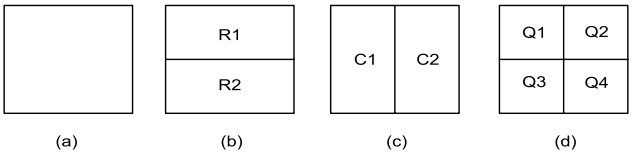


Fig. 8. Different partitions for 16×16 MB. (a) 16×16. (b) 16×8. (c) 8×16. (d) 8×8.

complex motion. For example, if an MB contains multiple objects that move in different directions, then using large partition (such as 16×16) for motion compensation will lead to large motion-compensated residual, which severely degrades the encoding efficiency.

Since cloud gaming provides us the unique advantage for conveniently calculating pixel-wise MVs (explained in Section IV-A), in this section, we propose a fast mode selection algorithm that exploits these MV characteristics to help select the appropriate block size. The basic idea is that by analyzing the homogeneity of the pixel-wise MVs within a 16×16 MB, we propose a set of candidate coding modes and eliminate unlikely modes in order to reduce the modes that need to be tested in RDO.

Inspired by the tree-structure block sizes defined in H.264/AVC [7], we propose four metrics to indicate the homogeneity for different regions within a 16×16 MB.

The first metric, $\mathrm{VAR}_{16\times16}$, is the variance of all the 256 pixels' MVs

$$\mathrm{VAR}_{16\times16} = \frac{1}{256}\sum_{i=1}^{16}\sum_{j=1}^{16}\left|\mathrm{PMV}_{i,j} - \frac{1}{256}\sum_{i=1}^{16}\sum_{j=1}^{16}\mathrm{PMV}_{i,j}\right|^2 \tag{23}$$

where i and j are the row and column indexes of a pixel within a 16×16 MB and $\mathrm{PMV}_{i,j}$ is the pixel-level MV for the pixel located at row i and column j.

The second metric, $\mathrm{VAR}_{16\times8}$, is defined as the average variance of the top half's and bottom half's pixel-level motion vectors. As shown in Fig. 8(b), we denote the top half of an MB by region R_1 and denote the bottom half by region R_2. Then, $\mathrm{VAR}_{16\times8}$ is computed as

$$\mathrm{VAR}_{16\times8} = \frac{1}{2}\sum_{k=1}^{2}\frac{1}{128}\sum_{(i,j)\in R_k}\left|\mathrm{PMV}_{i,j} - \frac{1}{128}\sum_{(i,j)\in R_k}\mathrm{PMV}_{i,j}\right|^2. \tag{24}$$

Similarly, the other two terms $\mathrm{VAR}_{8\times16}$ and $\mathrm{VAR}_{8\times8}$ are defined as the average variance of different regions' pixel-level MVs, and the corresponding region partitions are shown in Fig. 8(c) and (d)

$$\mathrm{VAR}_{8\times16} = \frac{1}{2}\sum_{k=1}^{2}\frac{1}{128}\sum_{(i,j)\in C_k}\left|\mathrm{PMV}_{i,j} - \frac{1}{128}\sum_{(i,j)\in C_k}\mathrm{PMV}_{i,j}\right|^2 \tag{25}$$

$$\mathrm{VAR}_{8\times8} = \frac{1}{4}\sum_{k=1}^{4}\frac{1}{64}\sum_{(i,j)\in Q_k}\left|\mathrm{PMV}_{i,j} - \frac{1}{64}\sum_{(i,j)\in R_k}\mathrm{PMV}_{i,j}\right|^2. \tag{26}$$

Metrics $\mathrm{VAR}_{16\times16}$, $\mathrm{VAR}_{16\times8}$, $\mathrm{VAR}_{8\times16}$, and $\mathrm{VAR}_{8\times8}$ are used to measure the homogeneity of the 256 pixel-wise MVs in different regions (higher value corresponds to lower homogeneity). We can use this information to estimate what mode will have the higher probability leading to a low RD cost. For example, for a 16×16 MB, if the metric $\mathrm{VAR}_{16\times8}$ is very low, then we can infer that the pixels in the top half [region R_1 in Fig. 8(b)] have homogenous motion and the pixels in the bottom half [region R_2 in Fig. 8(b)] have homogenous motion as well. Then, encoding regions R_1 and R_2 separately (encoding this MB using Inter_16×8 mode) may lead to small motion-compensated residuals and low RD cost; hence, Inter_16×8 is very likely to be selected as the optimal mode after the RDO process.

Inspired by the above analysis, we classify each 16×16 MB into one of the following categories when the specified condition is satisfied.

Category A: The entire 16×16 MB is homogeneous

$$\mathrm{VAR}_{16\times16} \leq T_c. \tag{27}$$

Category B: The motion within the 16×16 MB is complex and exhibits no obvious homogeneity in any subpartition

$$\begin{aligned} &\mathrm{VAR}_{16\times16} > T_c \quad \text{and} \quad \mathrm{VAR}_{16\times8} > T_c \quad \text{and} \\ &\mathrm{VAR}_{8\times16} > T_c \quad \text{and} \quad \mathrm{VAR}_{8\times8} > T_c. \end{aligned} \tag{28}$$

If the MB does not satisfy the above two conditions, it is further classified into one of the following categories.

Category C: The motion in the MB is more likely to be homogeneous within the top half and the bottom half, therefore it is suitable to be encoded using 16×8 partition

$$\mathrm{VAR}_{16\times8} < \mathrm{VAR}_{8\times16} \quad \text{and} \quad \mathrm{VAR}_{16\times8} \leq T_c. \tag{29}$$

Category D: The motion in the MB is more likely to be homogeneous within the left half and the right half, therefore it is suitable to be encoded using 8×16 partition

$$\mathrm{VAR}_{8\times16} < \mathrm{VAR}_{16\times8} \quad \text{and} \quad \mathrm{VAR}_{8\times16} \leq T_c. \tag{30}$$

Category E: The MB is more suitable to be encoded using 8×8 partition

$$\mathrm{VAR}_{8\times8} \leq T_c \quad \text{and} \quad \mathrm{VAR}_{16\times8} > T_c \quad \text{and} \quad \mathrm{VAR}_{16\times8} > T_c. \tag{31}$$

The value of the threshold T_C is selected to be 0.25 by extensive experiments. This value achieves a good and consistent performance on a variety of videos including different game scenes and activities.

Based on the above MB classification according to the pixel-level motion homogeneity, the candidate inter modes for each MB category are summarized in Table II (the prefix Inter_ is omitted).

By utilizing the motion homogeneity, we have established a heuristic to generate candidate inter modes. Further optimization can be achieved using the detection of uncovered pixels (explained in Section IV-A) to eliminate the inter modes for some MBs. This idea is inspired from the intuition that for the MBs that contain the uncovered pixels (namely, uncovered MBs), we cannot find the corresponding colocated MBs in the previous reference frame, and therefore there would be large motion-compensated residual if we encode these MBs using inter mode. Hence, we propose to directly bypass all the inter modes for all the MBs that contain uncovered pixels.

TABLE II
CANDIDATE INTER MODE FOR EACH MB CATEGORY

MB category	Candidate Inter-modes
A	16x16
B	16x16, 16x8, 8x16, 8x8
C	16x16, 16x8
D	16x16, 8x16
E	16x16, 8x8

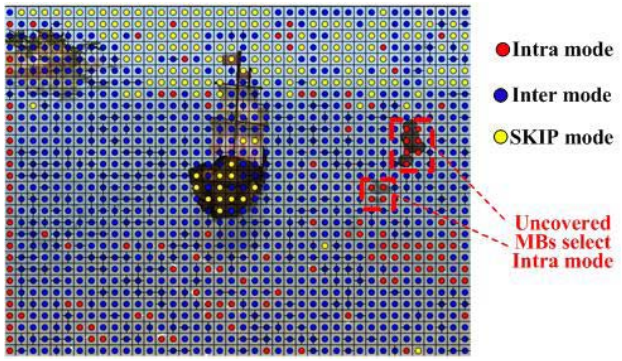


Fig. 9. Game frame for Broadsides and the optimal MB mode selected with full mode RDO.

Fig. 9 shows an example that supports the idea of eliminating all the inter modes for uncovered MBs. Fig. 9 shows the optimal modes for each MB selected by the regular H.264 encoder that performs RDO on all the possible modes (including intra modes, inter modes, and SKIP mode). Note that the game frame shown in Fig. 9 is identical to the frame shown in Fig. 7(b). We can observe that the uncovered MBs, which contain pixels that did not appear in the reference frame [shown previously in Fig. 7(c)], are mostly encoded using intra mode.

As a summary, the overall flowchart of the fast mode selection algorithm is shown in Fig. 10. To encode a 16×16 MB, we first test whether it satisfies the SKIP condition [28]. If yes, we just encode it with the SKIP mode. Otherwise, we then test if this MB contains uncovered pixels. If yes, then we claim that encoding this MB using inter mode is not suitable and we consider only intra-mode encoding. If the MB does not consider uncovered pixels, we first generate candidate inter-mode set according to Table II, and then consider both inter mode and intra mode as possible modes for RD cost calculation. Finally, we calculate the RD cost of each possible inter or intra mode and select the one with the minimum RD cost as the optimal mode.

Note that the proposed algorithm is only used for MB that belongs to P slice. For those MBs that belongs to I slice, it must be encoded using intra mode, and we just use the regular H.264/AVC encoding process to maintain high encoding quality for I slices. Furthermore, we assume that the B frame will be disabled since cloud gaming requires

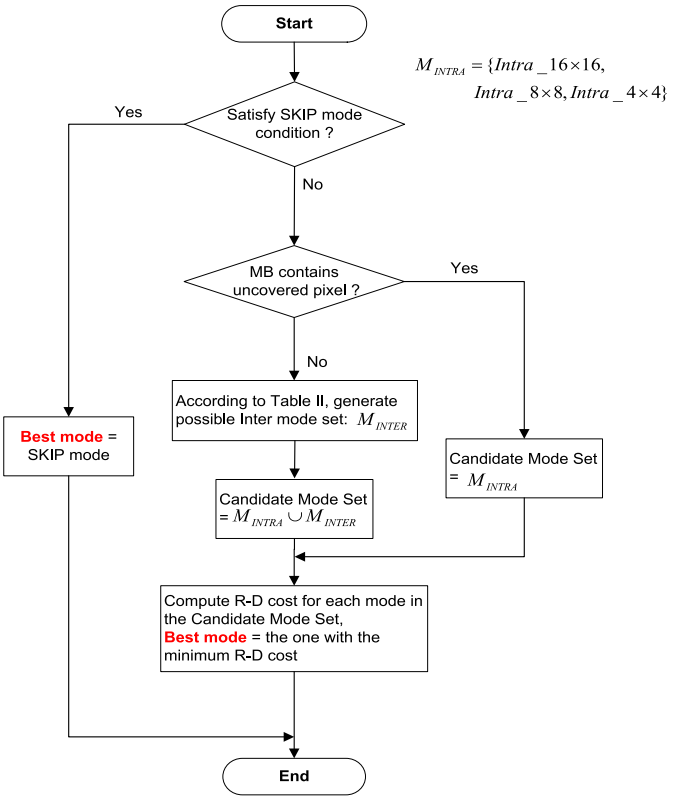


Fig. 10. Flowchart of the proposed fast mode selection algorithm.

real-time video encoding and has strict requirement about encoding delay, and hence the typical encoding group of pictures (GOP) structure is IPPP.

C. Assessment Experiment

In this section, we present the results of two rounds of assessment experiments, where we apply the proposed REA technique on a variety of game sequences. The first round of experiments is for studying the performance (speed and quality) of the proposed direct MV calculation method (explained in Section IV-A), and the second round of experiments is for studying the performance of the entire REA technique (including the direct MV calculation method and the fast mode selection algorithm).

We have captured six representative game videos as the source for the test videos: 1) four sequences for game PlaneShift and 2) two sequences for game Broadside. The video sequence name and descriptions are summarized in Table III. These six videos are selected such that our experiment has covered different game scene types and different motion properties.

To encode these video sequences, the following encoding setting is applied: GOP structure is IPPP, GOP size is set to be 30, video frame rate is 30 frames/s, video resolution is 800 × 600, reference frame number is set to be 1 (for low encoding delay), RDO and context-adaptive binary arithmetic coding are used in the main profile, and MV resolution is 1/4 pel. A group of QP values from 24 to 36 is used to encode. We use the open source X264 encoder software [27] as the implementation of the H.264/AVC standard. The version

TABLE III
DESCRIPTIONS OF EXPERIMENT VIDEOS

Video Sequence	Description
POH	Game PlaneShift, outdoor scene, high motion
POL	Game PlaneShift, outdoor scene, low motion
PIH	Game PlaneShift, indoor scene, high motion
PIL	Game PlaneShift, indoor scene, low motion
BH	Game Broadside, high motion
BL	Game Broadside, low motion

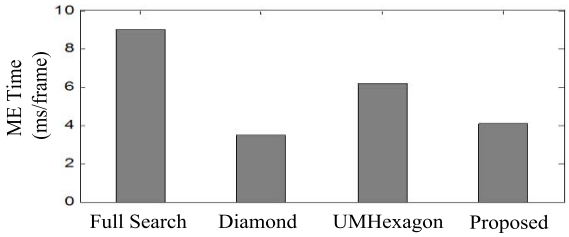


Fig. 11. ME time per frame using different ME approaches.

number of the X264 software was R2389. The cloud game server and the encoder are executed on a PC with Intel i7 2.7-GHz CPU and 8-GB RAM.

Next, we will discuss the procedures and results of two rounds of experiments. Both of these experiments use the video sequences and the encoding settings described above.

1) First Round of Experiments: First, we conducted an experiment to compare the computation time of our proposed direct MV calculation method with full search method and two classic fast ME methods: 1) Diamond search [29] and 2) UMHexagon search [30]. In order to compare these three ME methods with our proposed method, we configure the x264 encoder to use full mode selection, encode the video sequences listed in Table III, and record the time spent on motion estimation process. Note that the other three ME methods (full search, Diamond, and UMHexagon) are included in the X264 source code, so we do not need to implement them by ourselves.

From the results of ME time shown in Fig. 11, we can observe that full search method takes the most time, Diamond search takes least time, and our proposed method takes slightly longer than Diamond, but much less time than UMHexagon. The results tell us that although our proposed MV calculation method performs pixel-level calculation that seems to be computationally intensive, its actual complexity is similar to the Diamond search. The analytical explanation is shown in [31].

Second, in order to evaluate the quality degradation caused by the proposed direct MV calculation method (not by the fast mode selection), we conduct the second experiment, which is outlined as follows.

1) We first encode the six game videos listed in Table III of the revised manuscript using a regular x264 encoder, with full-search ME and full model selection. The MV resolution is 1/4 pel, and the MV search range is ±16 pels.

TABLE IV
AVERAGE PSNR DIFFERENCE FOR TEST VIDEOS

Video Sequence	POH	POL	PIH	PIL	BH	BL
$\Delta PSNR$ (dB)	-0.197	-0.078	-0.225	-0.086	-0.114	-0.065

2) We then encode the same videos again with a customized x264 encoder, in which we have modified the source code to use the MVs calculated with our approach to substitute the MVs derived with normal ME method. During this second encoding process, the customized encoder still uses full mode selection in order to make sure that the mode selection method is the same with the first encoding process.

Table IV lists the average peak signal-to-noise ratio (PSNR) difference between the two encoders for each test video sequence over a group of QP values from 24 to 36. The negative values in Table IV indicate that our proposed ME approach achieves lower PSNR than regular ME method. But as we can see, among the six test videos, the PSNR degradation varies between 0.065 and 0.225 dB. The small PSNR difference values demonstrate that the quality degradation of our proposed ME method is very slight compared with normal ME method.

As a summary, from Fig. 11 and Table IV, we can conclude that: 1) our proposed MV calculation method has a similar computational complexity with other fast ME methods such as Diamond search and 2) the proposed method has very limited quality degradation compared with normal ME (full search). However, the advantage of our proposed method is obvious: with the same complexity, it can produce pixel-level MVs, which provides more information than the regular block-level MVs, and can benefit the subsequent mode selection step.

2) Second Round of Experiments: In the second round of experiments, we study the performance (encoding speed and video quality) of the entire REA technique, including both the MV calculation method and the fast mode selection algorithm.

We first use the regular X264 encoder with UMHexagon [30] ME and full mode selection as the reference encoder. We then implement three fast encoding techniques, including our proposed REA technique, Fechteler's technique [17], and Liu's technique [32], respectively, for performance comparison.

Our proposed REA technique is different from Fechteler's and Liu's technique in the following aspects.

1) Fechteler's technique uses rendering information to compute MVs to bypass the ME process, but it does not optimize the mode selection process.
2) Liu's technique is a fast mode selection technique that utilizes the motion homogeneity to eliminate some unlikely block modes during the RDO process, and it can optimize the mode selection process, but it requires to perform block-level ME for every 4×4 block.

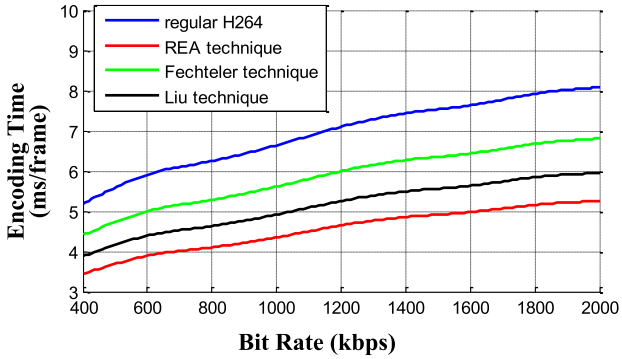


Fig. 12. Relation between encoding time per frame and bit rate for video sequence BOH.

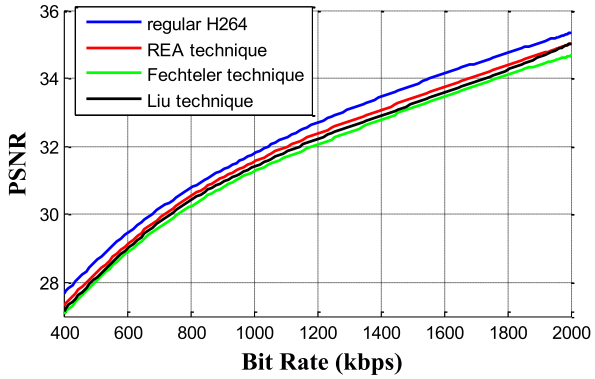


Fig. 13. Relation between PSNR and bit rate for the video sequence BOH.

3) Our technique first utilizes rendering information to bypass regular ME, and subsequently uses MVs' properties to optimize the mode selection process as well.

Figs. 12 and 13 show an example of the experimental results using video sequence POH. Fig. 12 shows the relation between encoding time per frame and the video bit rate. Fig. 13 shows the encoding efficiency: the relation between PSNR and bit rate. We can observe from Figs. 12 and 13 that our proposed approach can achieve much lower encoding time compared with the other three approaches, without noticeable video quality degradation (decrease in PSNR). Compared our proposed technique with Fechteler's technique, we can observe that our technique leads to a lower computation time (shown in Fig. 12) while keeping a higher encoding efficiency (shown in Fig. 13). This is because our proposed fast mode selection algorithm reduces the possible tested modes for the RDO process.

Furthermore, if we compare our proposed REA technique with Liu's fast mode selection technique, we find that our technique can also achieve a faster encoding speed while maintaining a slightly higher video quality. This is because of the following two reasons.

1) We have proposed in our technique that for those uncovered MBs (defined in Section IV-A), we will disable inter-prediction mode because for these blocks, there is no corresponding blocks in the reference frame and inter predication might lead to large motion-compensated residual. Liu's method [32], on the other hand, does not have this mechanism because in Liu's method, the encoder does not have any information about which

TABLE V
AVERAGE PERFORMANCE COMPARISON OF THE THREE FAST ENCODING TECHNIQUES

Video Sequence	Propose REA technique			Fechteler's technique		Liu's technique	
	BDPSNR	TS	Coverage	BDPSNR	TS	BDPSNR	TS
POH	-0.439	38.2%	83.2%	-0.635	22.8%	-0.465	31.6%
POL	-0.353	45.4%	90.7%	-0.603	22.1%	-0.357	33.0%
PIH	-0.382	36.9%	85.3%	-0.669	22.5%	-0.440	33.5%
PIL	-0.365	52.0%	91.9%	-0.595	23.5%	-0.378	35.2%
BH	-0.414	33.3%	88.7%	-0.604	20.8%	-0.434	31.8%
BL	-0.375	49.5%	93.7%	-0.596	23.8%	-0.415	33.6%
Ave.	-0.388	42.6%	88.9%	-0.617	22.3%	-0.415	33.1%

blocks belong to which objects and which objects are new emerging objects.

2) To encode a 16 × 16 MB, in our proposed technique, the mode selection heuristic is based on 256 pixel-level MVs, while in Liu's technique, it is based on 16 block-level MVs (each one corresponds to a 4 × 4 block). Given that the computation complexity for generating the MVs is approximately the same for our technique and Liu's technique, our technique provides fine-grained MV information, and hence may achieve a higher accuracy in mode selection and, therefore, higher encoding efficiency.

Table V shows the experimental results from a different perspective; it lists the results of all the six video sequences (for each video sequence, the average results over all the QP values are shown). To evaluate the average encoding performance (efficiency and complexity) for each video, we calculate Bjontegaard delta PSNR (BDPSNR) specified in [33] between each fast encoding technique (ours, Fechteler's, and Liu's) and the reference encoder (the regular X264 encoder). BDPSNR is a metric used to measure the average PSNR difference between two RD curves. Compared with the reference encoder, the average encoding time saving in percentage is defined as

$$\mathrm{TS} = \frac{T_r - T_f}{T_r} \times 100\% \tag{32}$$

where T_r and T_f indicate the encoding time per frame for reference encoder and fast encoder.

It can be observed from Table V that our proposed fast encoding technique can achieve on average of 42.6% reduction in encoding time compared with the regular X264 encoder over all the six game video sequences, at the cost of a PSNR degradation of only 0.388 dB on average. We can observe that the proposed fast encoding algorithm achieves a consistent gain in encoding time savings for all sequences with the least gain of 33.3% in video sequence BH and the greatest gain of 52.0% in video sequence PIL. Compared with Fechteler's algorithm, our proposed technique is faster and can reduce an extra 20.3% encoding time, and the PSNR degradation of our technique is 0.229 dB smaller than that of Fechteler's algorithm. Compared with Liu's algorithm, our proposed REA technique can reduce an extra 9.5% encoding time. Furthermore, the PSNR degradation of our technique is 0.027 dB smaller than that of Liu's technique.

In Table V, we have listed another metric called *coverage*, which is defined as the ratio of the number of MBs whose optimal encoding modes are covered by our proposed fast mode selection algorithm (Fig. 10) to the total number of MBs. We can observe that our proposed fast mode selection algorithm can make the optimal choice in 88.9% of all the MBs. This high accuracy of mode selection is the main reason that our proposed algorithm has only a 0.388-dB PSNR degradation compared with the regular H.264/AVC encoder with full search RDO.

Although the proposed REA technique can significantly reduce the encoding time, there exist one limitation: the encoding efficiency may drop if the game has complex lighting and rendering effect such as shadowing and reflection, since in these scenarios, for the same pixel, its value (colors and luminance) may not be the same in consecutive frames. If we use the collocated block in the previous frame as the reference block for inter prediction, the resulted motion-compensated residual might be large, leading to lowering of the encoding efficiency.

V. CONCLUSION

In this paper, we propose two rendering-based techniques that utilize rendering information to enhance video encoding for cloud gaming application to achieve: 1) better video perceptual quality under a fixed bit rate budget and 2) lower encoding complexity and encoding time for every video frame.

The rendering-based prioritized encoding technique exploits rendering information including depth map and scene composition information to prioritize different regions of a game frame, and adapts the QP values for each MB according to their importance and the bit rate budget. Assessment experiments have been carried out to validate the effectiveness of this prioritized encoding technique. The experimental results show that this technique can significantly increase the player's perceptual video quality under a given bit rate budget.

The proposed REA technique uses rendering information to reduce the computational complexity of the encoding process of H.264/AVC technique by:

1) directly calculating the motion vectors and eliminating the need for regular search-based motion estimation;
2) using rendering information to reduce the number of candidate encoding modes for ROD process.

The experimental results show that the proposed REA technique can achieve a significant and consistent encoding time reduction compared with regular H.264/AVC standard over a variety of test videos.

The two techniques proposed in this paper have utilized rendering information that can be very conveniently obtained, such as the depth map and object ID. In the future, in addition to using the above rendering information, we plan to investigate the use of pixel-domain features of the rendered video frame, such as luminance and chrome values of every pixel, to increase the perceptual video quality as well as

accelerate video encoding. Furthermore, these pixel-domain features can be used to detect the object boundaries or to estimate the spatial complexity of an MB, and hence can help in optimizing the mode selection algorithm by eliminating some unlikely intra modes. These are interesting topics that we plan to investigate in the future to further enhance the quality and efficiency of video encoding of cloud gaming.

REFERENCES

[1] *OnLive*. [Online]. Available: http://www.onlive.com, accessed Jul. 23, 2014.

[2] *Gaikai*. [Online]. Available: http://www.Gaikai.com, accessed Jul. 23, 2014.

[3] B. A. Wandell, *Foundations of Vision*. Sunderland, MA, USA: Sinauer Associates, Inc., 1995.

[4] S. Wang and S. Dey, "Cloud mobile gaming: Modeling and measuring user experience in mobile wireless networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 16, no. 1, pp. 10–21, Jan. 2012.

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

[6] M. Hemmati, A. Javadtalab, A. A. N. Shirehjini, S. Shirmohammadi, and T. Arici, "Game as video: Bit rate reduction through adaptive object encoding," in *Proc. 23rd ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, New York, NY, USA, 2013, pp. 7–12.

[7] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[8] 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.

[9] S. Shirmohammadi, "Adaptive streaming in mobile cloud gaming," in *Proc. IEEE COMSOC Multimedia Commun. Tech. Committee E-Lett.*, Sep. 2013, pp. 20–23.

[10] Y. Liu, Z. G. Li, and Y. C. Soh, "Region-of-interest based resource allocation for conversational video communication of H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 1, pp. 134–139, Jan. 2008.

[11] G.-L. Wu, Y.-J. Fu, and S.-Y. Chien, "Region-based perceptual quality regulable bit allocation and rate control for video coding applications," in *Proc. IEEE VCIP*, Nov. 2012, pp. 1–6.

[12] W. Lai, X.-D. Gu, R.-H. Wang, W.-Y. Ma, and H.-J. Zhang, "A content-based bit allocation model for video streaming," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jun. 2004, pp. 1315–1318.

[13] Z. Li, S. Qin, and L. Itti, "Visual attention guided bit allocation in video compression," *Image Vis. Comput.*, vol. 29, no. 1, pp. 1–14, Jan. 2011.

[14] H. Ahmadi, S. Z. Tootaghaj, M. R. Hashemi, and S. Shirmohammadi, "A game attention model for efficient bit rate allocation in cloud gaming," *Multimedia Syst.*, vol. 20, no. 5, pp. 485–501, Oct. 2014.

[15] N. Tizon, C. Moreno, and M. Preda, "ROI based video streaming for 3D remote rendering," in *Proc. IEEE 13th Int. Workshop Multimedia Signal Process. (MMSP)*, Hangzhou, China, Oct. 2011, pp. 1315–1318.

[16] M. R. H. Taher, H. Ahmadi, and M. R. Hashemi, "Power-aware analysis of H.264/AVC encoding parameters for cloud gaming," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, Jul. 2014, pp. 1–6.

[17] P. Fechteler and P. Eisert, "Accelerated video encoding using render context information," in *Proc. 17th IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2010, pp. 2033–2036.

[18] M. Semsarzadeh, M. Hemmati, A. Javadtalab, A. Yassine, and S. Shirmohammadi, "A video encoding speed-up architecture for cloud gaming," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, Jul. 2014, pp. 1–6.

[19] G. Cheung, A. Ortega, and T. Sakamoto, "Fast H.264 mode selection using depth information for distributed game viewing," in *Proc. IS&T/SPIE Vis. Commun. Image Process. (VCIP)*, San Jose, CA, USA, Jan. 2008.

[20] S. Shi, C.-H. Hsu, K. Nahrstedt, and R. Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," in *Proc. 19th ACM Int. Conf. Multimedia*, Nov. 2011, pp. 103–112.

[21] *PlaneShift*. [Online]. Available: http://www.planeshift.it/, accessed May 6, 2014.

[22] *Broadsides*. [Online]. Available: http://cse125.ucsd.edu/cse125/2012/cse125g1/, accessed May 6, 2014.

[23] B. Ciubotaru, G. Muntean, and G. Ghinea, "Objective assessment of region of interest-aware adaptive multimedia streaming quality," *IEEE Trans. Broadcast.*, vol. 55, no. 2, pp. 202–212, Jun. 2009.

[24] W. Ding and B. Liu, "Rate control of MPEG video coding and recording by rate-quantization modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 1, pp. 12–20, Feb. 1996.

[25] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using quadratic rate distortion model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 246–250, Feb. 1997.

[26] *Methodology for the Subjective Assessment of the Quality of Television Pictures*, document ITU-Rec. BT.500-11, 2002.

[27] *X264*. [Online]. Available: http://www.videolan.org/developers/x264.html, accessed Feb. 10, 2014.

[28] I. Choi, J. Lee, and B. Jeon, "Fast coding mode selection with rate-distortion optimization for MPEG-4 part-10 AVC/H.264," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 12, pp. 1557–1561, Dec. 2006.

[29] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[30] C. Zhu, X. Lin, L. Chau, and L.-M. Po, "Enhanced hexagonal search for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1210–1214, Oct. 2004.

[31] *Supplementary Material*. [Online]. Available: http://esdat.ucsd.edu/TCSVT_paper.html, accessed Feb. 10, 2014.

[32] Z. Liu, L. Shen, and Z. Zhang, "An efficient intermode decision algorithm based on motion homogeneity for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 1, pp. 128–132, Jan. 2009.

[33] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document VCEG-M33, Apr. 2001.

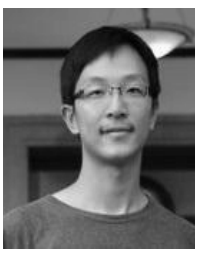
Yao Liu (M'10) is currently working toward the Ph.D. degree with University of California at San Diego, La Jolla, CA, USA.

His industry experiences include interning with Qualcomm, Beijing, China, in 2010, and Yahoo, Santa Clara, CA, USA, in 2013. His research interests include mobile multimedia, wireless communication, and mobile cloud computing.

Sujit Dey (SM'03–F'14) received the Ph.D. degree in computer science from Duke University, Durham, NC, USA, in 1991.

He was a Senior Research Staff Member with NEC Research Laboratories in Princeton, NJ, USA. He joined the University of California at San Diego (UCSD), La Jolla, CA, USA, in 1997. He is currently a Professor with the Department of Electrical and Computer Engineering, UCSD, where he is the Head of the Mobile Systems Design Laboratory, which is involved in developing innovative mobile cloud computing architectures and algorithms, adaptive multimedia, and networking techniques to enable the next generation of mobile multimedia applications. He is also the Director of the UCSD Center for Wireless Communications. He also serves as the Faculty Director of the von Liebig Entrepreneurism Center, and is affiliated with the Qualcomm Institute. He founded Ortiva Wireless in 2004, where he served as its founding CEO and later as CTO till its acquisition by Allot Communications in 2012. He has co-authored over 200 publications, including journal and conference papers, and a book on low-power design. He is the coinventor of 18 U.S. patents, resulting in multiple technology licensing and commercialization.

Dr. Dey has been a recipient of the six IEEE/ACM Best Paper awards, and has been the Chair of the multiple IEEE conferences and workshops.



Yao Lu (M'12) received the B.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 2012. He is currently working toward the Ph.D. degree with University of California at San Diego, La Jolla, CA, USA.

His research interests include mobile multimedia, computer graphics, video encoding, computer networks, and cloud computing.